A review on feature selection in mobile malware detection

Article *in* Digital Investigation · March 2015

DOI: 10.1016/j.diin.2015.02.001

4 authors:

Ali Feizollah
University of Malaya
8 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Nor Badrul Anuar
University of Malaya
112 PUBLICATIONS **970** CITATIONS
SEE PROFILE

Rosli Salleh
University of Malaya
71 PUBLICATIONS **321** CITATIONS
SEE PROFILE

Ainuddin Wahid
University of Malaya
60 PUBLICATIONS **179** CITATIONS
SEE PROFILE

Digital Investigation 13 (2015) 22–37

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

A review on feature selection in mobile malware detection

Ali Feizollah*, Nor Badrul Anuar, Rosli Salleh, Ainuddin Wahid Abdul Wahab

Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, 50603, Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 11 October 2014
Received in revised form 2 February 2015
Accepted 4 February 2015
Available online

Keywords:
Mobile malware
Android
Feature selection
Review paper
Mobile operating system

ABSTRACT

The widespread use of mobile devices in comparison to personal computers has led to a new era of information exchange. The purchase trends of personal computers have started decreasing whereas the shipment of mobile devices is increasing. In addition, the increasing power of mobile devices along with portability characteristics has attracted the attention of users. Not only are such devices popular among users, but they are favorite targets of attackers. The number of mobile malware is rapidly on the rise with malicious activities, such as stealing users data, sending premium messages and making phone call to premium numbers that users have no knowledge. Numerous studies have developed methods to thwart such attacks. In order to develop an effective detection system, we have to select a subset of features from hundreds of available features. In this paper, we studied 100 research works published between 2010 and 2014 with the perspective of feature selection in mobile malware detection. We categorize available features into four groups, namely, static features, dynamic features, hybrid features and applications metadata. Additionally, we discuss datasets used in the recent research studies as well as analyzing evaluation measures utilized.

Introduction

The ubiquity of mobile devices is undeniable because they have brought new possibilities to every days life. Contemporary mobile devices are more powerful when compared to Personal Computers (PCs) ten years ago. Unlike PCs, portability of mobile devices makes them attractive to users. In addition, their small sizes as compared to personal computers play an important role in increasing their popularity. Furthermore, users interests are increasing towards the Rich Mobile Applications (RMA), such as Google Maps that deliver rich user experience along with high interaction (Knoernschild, 2010). However, such popularity has serious security and privacy threats and various other malicious activities. The malicious activities are hidden from the user and are committed in the background or at midnight when the user is asleep (Eslahi et al., 2012). Based on such characteristics, we assess the research works done to detect these malware.

The aim of this paper is to scrutinize various features available in Android malware, since feature selection has considerable effects on results of experiments. We discuss such effect in the following sections. Suarez-Tangil et al. (2013) discuss malware for smart devices in general. However, the paper discusses various types of features very briefly and the authors did not cover all types of available features. Similarly, La Polla et al. (2013) investigates various types of mobile devices, available malware, their effect on the devices and different detection methods. Nevertheless, they did not mention what features they used in detection, considering that features have significant impact on detection. Mohite and Sonar (2014) survey different analysis techniques in mobile malware detection. The paper

* Corresponding author.
E-mail addresses: ali.feizollah@siswa.um.edu.my (A. Feizollah), badrul@um.edu.my (N.B. Anuar), rosli_salleh@um.edu.my (R. Salleh), ainuddin@um.edu.my (A.W.A. Wahab).

http://dx.doi.org/10.1016/j.diin.2015.02.001
1742-2876/

mention examples of detection methods along their description. The paper does not include datasets and evaluation measures. In addition, it does not cover all the recent works comprehensively. Peng et al. (2014) examines evolution of mobile malware, their damages and their propagation model. They included various operating system in the paper, which makes it difficult to examine all available aspects thoroughly. However, we focus on Android operating system and the results are more accurate and comprehensive. Additionally, to the best of our knowledge, surveying Android features is unprecedented in research works.

The rest of this paper is organized as follows. Section 2 gives background information needed for the rest of the paper. Section 3 examines four types of features in mobile malware detection including the static, dynamic, hybrid and applications metadata. We comprehensively analyze each type of features. Section 4 presents discussions regarding the datasets used in the recent research works and their description. Additionally, we discuss the evaluation measures of malware detection in this section. Finally, Section 5 concludes the paper by highlighting important points.

Background

In this section, we present background information. First, we investigate the supremacy of mobile devices and see where mobile devices stand against PCs. Next, we examine how widespread mobile malware are; we then explain different types of malware, ranging from simple one to the most dangerous and sophisticated one. It is beneficial to know popularity of mobile devices, as well as, the seriousness of mobile malware. We scrutinize the importance of feature selection in malware detection in the next sub-section in order to establish the necessity of this research work. Finally, we take a closer look at Android files and their components, since we refer to various components throughout this work.

Supremacy of mobile devices

Popularity of mobile devices is on the rise (Chen and Bilton, 2014). Gartner, an American information technology research and advisory firm, reported that total shipment of mobile devices increased in 2013 by 5.9% and reached 2.35 billion devices compared to the previous year and it is estimated that the growth continues to 2.5 billion devices in 2014 (Gartner, 2013). On the other hand, the shipment of PCs has declined to 305 million units in 2013 and it is expected to decrease below 300 million units in 2014 (Gartner, 2013). Table 1 shows the number of devices shipments in 2012, 2013 and 2014.

Table 1
Worldwide devices shipments (thousands of units) (Gartner, 2013).

	2012	2013	2014
PC (Desktop and Notebook)	341,273	305,178	289,239
Ultramobile	9787	20,301	39,824
Tablet	120,203	201,825	276,178
Mobile Phone	1,746,177	1,821,193	1,901,188
Total	2,217,440	2,348,497	2,506,429

The comparison between PC and mobile devices, ultramobile, tablets, and mobile phones, reveals that the number of PCs is decreasing while the shipment of mobile devices is increasing. In terms of usage of mobile devices, Walker Sands published a report that indicated the Internet traffic pertaining to mobile devices increased. Based on the report, the Internet traffic of mobile devices represents 67% increase in the third quarter of 2013 compared to the same period in 2012 (Sands, 2013).

The rise of android malware

There are numerous mobile operating systems in the market namely, Android (Google, 2014a), iOS (Apple, 2014), Windows Phone (Microsoft, 2014), and BlackBerry (R. in Motion, 2014). Android has dominated the mobile devices industry. Based on a report, a total of 261.1 million devices were shipped in the third quarter of 2013 and 81.3% of the shipped devices were running Android operating system (CNET, 2013). Fig. 1 depicts the dominance of Android among other mobile operating systems.

The number of attacks is steadily going up for Android. Based on the report from F-Secure, Android incorporated 79% of all malware in 2012 compared to 66.7% in 2011 and just 11.25% in 2010 (Techcrunch, 2013). Similarly, Symantec said that number of Android malware increased almost four times between June 2012 and June 2013 (Symantec, 2013). In addition, the period of April 2013 to June 2013 witnessed a massive growth of almost 200% in Android malware. Fortinet (2014), a world leader in high performance network security, announced that within the period of January 1, 2013 until December 31, 2013, they discovered over 1800 new distinct families of malware and the majority of which were Android malware. Malware growth not only degrades performance of the devices, but also has posed serious concerns towards the privacy and security of data (Fortinet, 2014). In February 2014, Symantec stated that an average of 272 new malware and five new malware families are discovered every month targeting Android operating system (Symantec, 2014).

The reason of such enormous increase in Android malware lies in the fact that Android is an open source operating system (Teufl et al., 2013) and the application market of Android, known as Google Play, is not monitored meticulously in terms of security (Feizollah et al., 2013).

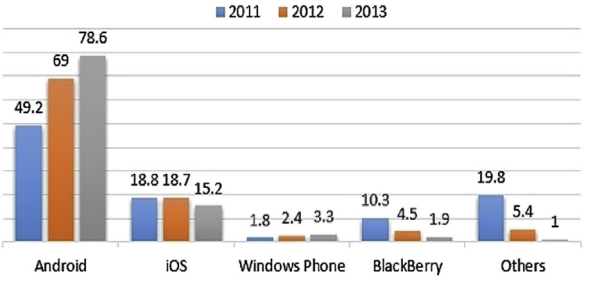

Fig. 1. Mobile operating systems market share (% of global unit shipments) (Motley, 2014).

Moreover, there are some unofficial Android markets, for example Aptoide (2014) in which the security issues are not paid much attention. Furthermore, as mentioned earlier, the market share of Android among mobile operating systems is considerably high. Consequently, attackers target Android to gain more benefits as compared to other operating systems.

Types of android malware

There are variety of attacks particular to Android ranging from adware to the most sophisticated and dangerous ones. Adware have the purpose of just advertising a product or a website that are harmless but annoying. The most dangerous and sophisticated malware are capable of accessing personal data on the device as well as hijacking the mobile device.

Android Dowgin is an example of an adware that installs itself on Android device as a bundle with other application. It then displays advertisements in the notification area of the device and is not easily removed. It is estimated that between 10,000 and 50,000 users are infected with this adware (AVG.ThreatLabs, 2013). It has been spreading since July 2013 and continues to proliferate (Eset, 2013). The alarming issue is that as of December 2013, some of the prominent anti-virus software such as Symantec, Trend-Micro, and McAfee were not able to detect it (Virustotal, 2013).

The Android attackers, sometimes, have financial encouragements and also have turned out to be more aggressive recently (Symantec, 2014). Upon installation, some applications send expensive short message service (SMS) to premium numbers without users knowledge that reflects itself in users bills. Such applications have been on the rise for years. A report published in 2013 shows that some attackers earn up to USD 12,000 per month via such malware (The.Register, 2013). Fig. 2 shows the number of malware with financial motivations within the period of 2006 and 2012.

Based on a report by Sophos, a security firm, a malicious version of the popular Angry Bird game secretly sends premium SMS that costs GBP 15. Each time the user starts the application, it sends a premium SMS. It is estimated that 1391 devices were infected with this malware and it is evaluated that the developers of this malicious application earned GBP 27,850 through sending SMS to premium numbers (Sophos, 2012).

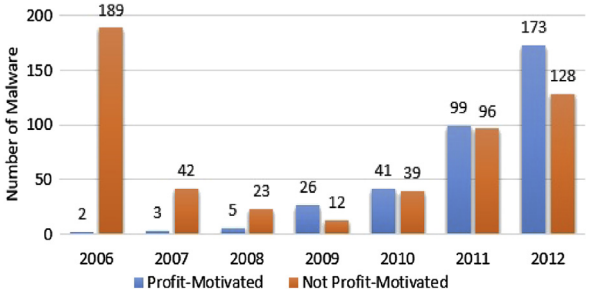

Fig. 2. Number of mobile malware motivated by profit per year, 2006–2012 (Techcrunch, 2013).

Some of malware have gone further by making phone calls in the background without users knowledge. MouaBad is a malware that secretly makes phone calls in such a way that it waits until a while after the devices screen goes off and the screen becomes locked. It then starts calling premium numbers. As soon as the user interacts with the device, the malware ends the call. Fortunately, this malware does not alter call logs and the user is able to discover existence of the malware by checking the call logs (Lookout, 2013).

A botnet is more dangerous than the aforementioned malicious applications. Upon infecting the device, the attacker gains access to the devices and can perform malicious activities by controlling applications on the device. A botnet is the network of such infected devices. An attacker can access hundreds of infected devices in a single botnet (Vural et al., 1007). For example, security analysts discovered an infected version of the Angry Birds Space in April 2012. It functions like a normal application without suspicious symptoms. However, it uses a software trick known as GingerBreak to acquire root access that allows it to do tasks outside its privilege. It secretly downloads malicious codes from a server and opens a back door for attackers upon which the device joins the botnet eventually (Sophos, 2013). The ZeroAccess botnet is adding approximately 100,000 new infections weekly by paying considerable amount of money weekly to generate new associated infections. It had 88.65% share of botnet dominance in 2013 (Fortinet, 2014).

Recently, attackers have adopted a new approach towards infecting mobile devices. Thus far, attackers were depending on alluring users to download their malicious applications after which the application performs malicious activity without users knowledge. It has been observed that personal computers have been used as a conduit to Android devices, which is called hybrid threats (Symantec, 2014). Trojan Droidpak uses hybrid threats to infect mobile devices. It first gains access to a personal computer and based on that a malicious Android application package (APK) file downloads itself. When the user connects an Android device to the computer, the malicious file attempts to install itself on the device. After the successful installation, it attempts to convince the user to download and install the infected version of Korean banking application (Symantec, 2014).

The importance of feature selection

Numerous studies try to confront the danger of Android malware. One of the first approaches is signature-based detection in which the detection system constructs a unique signature for a malware and detects malware by matching the signature with the collected data. However, a small modification in a malware leads to a new variant of the malware and is able to bypass the signature detection method (Garcia-Teodoro et al., 2009). DroidAnalytics (Zheng et al., 2013) was developed based on the signature-based method. It automatically collects, extracts, and analyzes the signature of an Android application file. It uses Java code and classes as a signature to detect the known malware. Nevertheless, it is unable to detect unknown

malware. With the steady rise of new variants of Android malware, it is vital to detect them effectively.

Researchers turned to machine learning methods to overcome the limitations of the signature-based method. We train machine learning algorithms based on collected data. They are capable of detecting anomalies, which is necessary to discover new malware. For example, Sahs and Khan (2012) performed an experiment in which they applied support vector machine (SVM) algorithm on their dataset. The authors trained the algorithm and achieved 93% of accuracy. Similarly, Garcia-Teodoro et al. (2009) presented a study in which they used k-means algorithm for malware detection. The authors achieved 87.39% of detection accuracy.

In machine learning methods, feature selection is one of the first and most crucial steps. In case of Android applications, they consist of various elements such as permissions, Java code, certification, behavior of the application on the device and its behavior on the network. Selecting the most useful subset of features from massive number of available features changes the result of the whole experiment (Guyon and Elisseeff, 2003). Some of the benefits of feature selection are as follows:

- Feature selection makes it possible to reduce dimensionality of datasets because with less data, it is possible to easily visualize the trend in data (Crussell et al.).
- Analyzing datasets involve processing vast amount of data and therefore, reducing them to a useful subset not only saves the time and cost of experiments, but also minimizes the time for real world implementation (Crussell et al.). Selecting useful subset of the features considerably reduces runtime of the machine learning algorithms in training phase.
- Feature selection removes noisy and irrelevant data from datasets leading to more accurate results of machine learning algorithms (Jensen and Shen, 2008).

We conducted two experiments to examine the effect of features on results. We collected network traffic of over 800 Android applications, including normal and malicious, from MalGenome (Yajin and Xuxian, 2012) data sample. The dataset consists of ten network traffic features out of which we selected five features for each experiment. The dataset comprises of 504,148 records. K-nearest neighbors classifier with three as the number of neighbors were used. Table 2 shows results of the experiments.

Table 2
Results of the experiments.

	Experiment 1	Experiment 2
Features	frame.len frame.number frame.time_delta frame.time_relative tcp.srcport	tcp.dstport tcp.window_size value tcp.seq ip.src ip.dst
True Positive Rate (TPR)	98.63%	99.98%
False Positive Rate (FPR)	1.37%	0.02%

As Table 2 illustrates, different features yield different results despite the fact that the data collection process and used classifier are same for both the experiments. Thus, the effect of feature selection is conspicuous. In addition, selection of the most useful features is an important and challenging task.

In this paper, we are scrutinizing various types of features available in Android applications and their trend in the research works. Moreover, we provide some guidelines on how to choose the best features. Additionally, we discuss various utilized datasets and present discussion on various evaluation measures employed by recent works.

Structure of android application package (APK)

Throughout this study, we refer to various components of an Android installation file, known as APK. It is beneficial to explain about different parts of such files. It is worth noting that an APK is an archive file type that softwares such as WinZip are able to open it (Sanz et al., 2012). Components of an APK file are as follows:

- AndroidManifest.xml: An XML file holding meta information about an application, such as descriptions and security permissions. Prior to installation of an Android application, the application provides prospective users with a list of permissions that are available in the file.
- Classes.dex: It contains source code of an application written in Java and compiled for Android that the machine converts it to a special file format with .dex extension.
- Resources: It entails all resources an application needs to run, such as pictures used in the application, layout of the application, its appearance to a user, use of a database, and data stored in the database.

Feature selection in mobile malware detection

An Android application consists of several parts that have the potential to be a feature in Android malware detection. Fig. 3 shows various types of features and subtypes of each category. In this work, we analyzed 100 papers, from highly credible sources such as IEEE and Springer, with the perspective of feature selection in Android malware detection. We selected publications between 2010 and March 2014. Based on our findings, researchers published 49 research papers regarding Android malware detection in year 2013. It signifies that the interest towards developing effective systems for Android malware detection is increasing compared to 22 papers in 2012 and 7 papers in 2011. Moreover, the first quarter of 2014 has 15 publications.

We have divided mobile malware features into four groups. Table 3 shows these four groups of features with their descriptions.

Additionally, we have analyzed recent works based on type of features they used. Fig. 4 shows the percentage of each group in reviewed papers. Based on Fig. 4, researchers chose static features nearly equal as dynamic features. Although the hybrid features are more

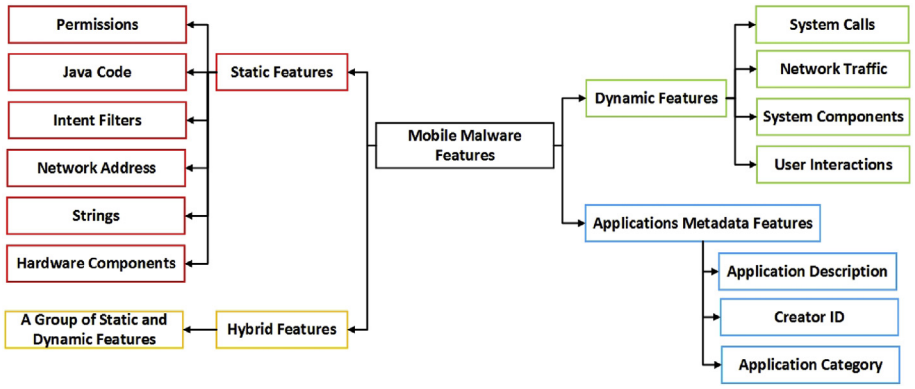

Fig. 3. Taxonomy of mobile malware features.

comprehensive, they constitute only 10% of the literatures. The apparent reason is the complicated process of using two types of features, since the hybrid features require collecting static and dynamic features separately. In the following sections, we discuss each type of features in detail.

Static features

Static features include features available in the apk file such as AndroidManifest.xml file and Java code file. Out of 100 papers reviewed, 45 papers used static features to conduct their experiments. Among the static features, researchers used Android permission in 36% of the papers (Yerima et al., 2014; Aung and Zaw, 2013; Grace et al.; Peng et al., 2012; Grace et al., 2307; Sarma et al.; Samra et al., 2013; Yerima et al., 2013; Sahs and Khan, 2012; Sanz et al., 2013a; Zhou et al.; Sanz et al., Bringas; Seo et al., 2014; Wu et al., 2012; Sanz et al., 2013b; Arp et al., 2014), more than other static features. Selection of Java code comes second with 29% of papers (Desnos, 2012; Rastogi et al., 2014; Faruki et al.; Suarez-Tangil et al., 2014; Grace et al.; Lu et al., 2012; Crussell et al.; Deshotels et al.; Zhou et al.; Shabtai et al., 2010a; Zheng et al., 2013; Almohri et al.; Zheng et al.). The following sections discuss the static features in details.

Table 3
Categorization of mobile malware features.

Type of feature	Description
Static	They are pertaining to the content of APK files. There are various features in APK files, such as permissions, Java code, intent filters, network address, strings and hardware components.
Dynamic	They represent post-installation behavior of applications on mobile devices and include behavior of the application in the operating system or on the network.
Hybrid	Hybrid features are combination of both static and dynamic features. They are the most comprehensive features because they analyze applications from various aspects.
Applications' Metadata	The last group consists of metadata pertaining to Android applications, such as their information on Google Play.

Android permission feature

We know that Android operating system has Linux core; among which it comprises important part of Linux security architecture. Prior to installation of an application, it provides list of requested permissions to the user. Upon granting the permissions by the user, the application installs itself on the device. There are 134 official Android permissions in Android 2.2 (i.e. API 8) (Felt et al.). Google categorized them into four groups, namely, normal, dangerous, signature, and signature or system (Google, 2014b). There are different approaches taken by researchers in analyzing Android permissions. Authors of (Peng et al., 2012; Wang et al., 2013; Pandita et al.; Grace et al.) used permissions to evaluate applications and ranked them based on possible risks (using probabilistic generative models, quantitative security risk assessment). Numerous studies simply extracted permissions and utilized machine learning to detect malicious application, (Samra et al., 2013; Aung and Zaw, 2013; Yerima et al., 2014; Sanz et al., 2013a). Researchers in (Moonsamy et al., 2013; Huang et al.) argue that merely analyzing requested permissions is not sufficient for detecting malicious applications. They analyzed used permissions in addition to

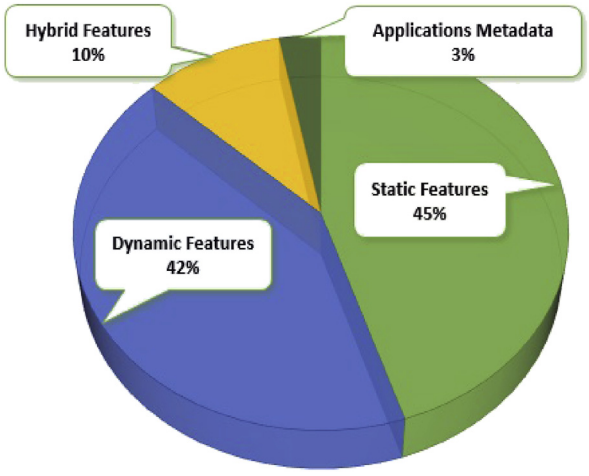

Fig. 4. Categorizing literature based on type of features.

requested permissions in order to detect malware. AppGuard (Backes et al.) has gone one step further and has extended Android permission system to alleviate current vulnerabilities. They claim that their system is a practical extension for Android permission system as it is possible to use it on devices without any modification or root access.

Why Android permission is the most used static feature? As mentioned earlier, Android operating system has Linux architecture. Permission is the first barrier to attackers. Even though the Java code contains malicious code, some of API calls in the code need permission to be invoked (Wu et al., 2012). Permission-protected API calls are part of the security features of Android operating system. For example, before sending a message or accessing the camera, Android checks if the application has permission to do so (Felt et al., 2046). Based on such scenario, focus of researchers is on permissions more than other static features to detect malware based on demanded permissions.

Android Java code feature

Developers write Android applications in Java programming language and subsequently compile them to a special format called Dalvik, which is proprietary to Android operating system. Google introduced Android runtime (ART) in the 4.4 release. Although ART offers new features (i.e. ahead-of-time compilation, improved garbage collection, development and debugging improvements), Dalvik remains the default runtime in the Android operating system (Google, 2014c). Researchers have used various analysis approaches on the Java code. Some researchers use Application Programming Interface (API) calls to detect malware (Rastogi et al., 2014; Grace et al., 2307; Deshotels et al.; Yerima et al., 2013; Zheng et al., 2013). Every Android application needs to have API calls to interact with the device. As an example, there are API calls to the telephony manager of the operating system to retrieve phone ID and subscriber ID. The API calls in a method are sequential. Researchers consider such a sequence as the applications signature that is unique to that application. Changing the sequence of API calls is a strategy used by attackers to bypass the detection process, called code obfuscation. Analyzing control flow of the Java code is another approach adopted by researchers (Suarez-Tangil et al., 2014; Crussell et al.; Xu et al., 2013; Chin et al., 2000; Sahs and Khan, 2012). Although attackers can change the sequence of API calls or rename API calls to evade detection system, the flow of the Java code does not change and researchers use it to develop stronger detection systems.

Other static features

Besides permissions and Java code, some researchers analyze several other static features that are as follows.

- ***Intent Filter***: Intent filter is one of the elements described in the manifest file. It is an abstract information about an operation, with which we infer intentions of the applications. For example, pick a contact, take a photo, dial a number, etc. Based on intent filters, Android takes appropriate actions. Researchers have been using intent filters for malware detection, since attackers command malware to send private data to them that requires the presence of intentions in the intent filter part of AndroidManifest.xml file.

In DroidMat (Wu et al., 2012), various features from an Android file including intent filters are extracted and are analyzed. The authors utilized several machine learning algorithms such as k-means, k-nearest neighbors and naive bayes to develop malware detection system. Evaluation of DroidMat exhibited an improvement over similar systems in that time.

Zhang et al. (Luoshi et al.) published a system (named A3) that considers several features including intent filters in Android installation file. It then constructs a call graph that represents flow of the Java code execution. It then uses A* algorithm to determine the shortest path that subsequently shows the behavior of the malware.

DREBIN (Arp et al., 2014) presents a broad static analysis. The approach collects static features of Android installation file including intent filters. They used support vector machine (SVM), a machine learning algorithm, for detection purpose. The results of the experiment showed that DREBIN detected 94% of malware with low false alarm.

- ***Network Address***: Attackers instructs malware to contact them and report their status or send users personal data. To do so, attackers embed address of the server, known as command & control (C&C) server, in malicious code of the malware. Researchers look for network address or IP address of the C&C server in code of Android installation files. Zhang et al. (Luoshi et al.) and Arp et al. (Arp et al., 2014) incorporated the network address as one of the static features in their systems.
- ***Strings***: Sanz et al. stated that one of the widely used techniques in classic malware detection is analyzing strings available in the file. They applied the same technique for Android malware by extracting every printable string in Android file, such as menus in the application or the server address with which the application connects. The authors used Vector Space Model (VSM) (Baeza-Yates and Ribeiro-Neto, 1999) to represent the strings as vectors in multidimensional space. Afterward, authors used distance measures, such as Manhattan distance, Euclidean distance and Cosine similarity to calculate anomaly of the data. The authors evaluated the results with 666 samples of Android applications. They achieved accuracy of 83.51% and TPR of 94% in the experiments.
- ***Hardware Components***: DREBIN (Arp et al., 2014) used hardware components as a static feature. As part of AndroidManifest.xml file, applications request combinations of hardware that they need in order to function, for example, the camera or GPS. Combinations of requested hardware imply harmfulness of the application, for example, 3G and GPS access implies a malware that reports location of the user to the attacker.

Other than mentioned static features, additional features have the potential to be used as Android static feature. Shabtai et al. (2010a) used features like .apk size, number of zip entries, number of files for each file type,

count of XML elements and features for each element name.

Dynamic features

We define dynamic features as behavior of the application in interaction with operating system or network connectivity. There are two main types of dynamic features used in recent works: system calls and network traffic. Every application demands resources and services from operating system by issuing system calls, such as read, write and open.

Network traffic is another dynamic feature used by researchers. Applications tend to connect to network to send and receive data, receive updates, or maliciously leak personal data to attackers. Monitoring network traffic of mobile devices is a way of catching culprit in the act. Based on our analysis, 42 out of 100 papers used dynamic features. Twenty-two papers used system calls as their dynamic feature and 10 papers used network traffic. The remaining 10 papers selected other dynamic features, such as system components or user interaction.

Android system call feature

There are more than 250 system calls in a Linux kernel that are also available in Android (Burguera et al., 2046). Analyzing system calls leads to anomaly detection in applications behavior (Feizollah et al., 2013). Applications use system calls to perform specific tasks such as read, write and open since they cannot directly interact with the Android operating system. Upon issuing a system call in user mode, Android operating system switches to kernel mode to perform the required task. System call is the most selected feature among dynamic features, constituting more than half of the reviewed papers. Works such as (Burguera et al.; Zhao et al.; Yan and Yin; Su et al., 2012; Khune and Thangakumar, 2012) captured and analyzed system calls to detect malicious application.

Android network traffic feature

Majority of applications, normal or malicious, require network connectivity. In (Yajin and Xuxian, 2012), the authors stated that 93% of their collected Android malware samples need network connection in order to connect to attackers. Additionally, Sarma et al. published a research work in 2012 in which they analyzed permissions of Android files. They examined over 150,000 applications and found out that 68.50% of normal applications require network access while 93.38% of malicious applications require network access. Similarly, in (Yerima et al., 2014) permissions of 2000 applications were analyzed. Over 93% of malicious applications requested network connectivity. It is evident that majority of applications request network access, particularly the malicious ones. Therefore, it behooves researchers to focus on analyzing network traffic for effective Android malware detection.

Despite the effectiveness of network traffic feature in mobile malware detection, it has not attracted researchers attention as much as the other dynamic features. Utilizing network traffic imposes the challenge of dealing with massive number of network records in the dataset that could be as many as a million records. Furthermore, analyzing collected network traffic requires profound understanding of network architecture.

Other dynamic features

In addition to system calls and network traffic, researchers have been using other dynamic features. The following discusses other dynamic features.

- ***System Components***: Mobile devices have similar components as personal computers, such as CPU and memory. Some researchers investigated detection of Android malware using system components. In MADAM (Dini et al., 1007), the authors analyzed CPU usage, free memory, and running processes of mobile devices that are considered kernel level of the operating system. In addition, it examined user/application level features, such as Bluetooth and Wi-Fi status of the device. The collected data were used to train k-nearest neighbors algorithm.

STREAM (Amos et al., 2013) was introduced in 2013 for Android operating system. It collects data regarding system components like cpuUser, cpuIdle, cpuSystem, cpuOther, memActive, and memMapped. It subsequently uses machine learning algorithms to train the system in order to detect Android malware. Ham and Choi (Hyo-Sik and Mi-Jung, 2013) and Hoffmann et al. (Hoffmann et al., 2013), also used system components as dynamic features.

- ***User Interaction***: Users are potential victims of malicious applications. Analyzing users interaction with applications is one of the possible solutions in malware detection. PuppetDroid (Gianazza et al., 2014) captures users interaction with the device (e.g. pushing a button, zooming and navigating through pages). The authors evaluated the system with 15 Android applications. The goal is that after capturing user interactions related to a malware, the system looks for similar user interaction to detect malicious applications.

Dynodroid (Machiry et al., 2491) is another system developed based on the user interaction analysis. It collects users activities, such as tapping the screen, long pressing and dragging. The evaluation of Dynodroid involves analyzing 50 Android applications. The results found bugs in Android applications.

As we mentioned in the static features section, intent filters are extracted from AndroidManifest.xml file as static feature. There is a potential to use intents as dynamic feature. Feng et al. (Feng et al., 2013) used intents as dynamic feature by monitoring them at run time.

Hybrid features

We define hybrid features as a group of static and dynamic features that are used together in detection systems. They are the most comprehensive features, since they involve vetting Android application installation file as well as analyzing behavior of the application at runtime. Blasing

et al. (Blasing et al., 2010) developed AASandbox which analyzes static and dynamic features. It extracts permissions and Java code from the APK file and uses them as static features. It then installs the application; logs system calls, and uses it as dynamic feature. Wei et al. (Wei et al.) published ProfileDroid in which they examined AndroidManifest.xml and Java code as static features. They chose user interaction, system calls and network traffic as dynamic features. Some of the similar works that chose hybrid features are as follows, (Zhou et al.; Spreitzenbarth et al.; Eder et al., 2013; Xu et al., 2013).

Android applications metadata

A few researchers opted to utilize Android applications metadata for malware detection. We define metadata as the information users see prior to the download and installation of the applications, such as the applications description, their requested permissions, their rating and information regarding developers. Applications metadata cannot be categorized as static or dynamic features as they have nothing to do with applications themselves.

In WHYPER (Pandita et al., Xie), the authors access the permissions requested by applications in the market and used Natural Language Processing (NLP) to look for sentences that justifies the need for the requested permissions. It achieved 82.8% precision for three permissions (address book, calendar and record audio) that protect sensitive and personal data.

Similarly, Teufl et al. (2013) used sophisticated knowledge discovery process and lean statistical methods to analyze the metadata gathered from Google Play. The authors argue that metadata analysis should be part of static or dynamic analysis as a complement. They collected the following data including the last time modified, category, price, description, permissions, rating, and number of downloads. The authors mentioned that the following data can also be used as metadata, such as creator ID, contact email, contact website, promotional video, number of screenshots, promo texts, recent changes, ID, package name, installation size, version, application type, ratings count and application title. The authors also used machine learning algorithms in their experiments. Definitions of some of the aforementioned metadata are as follows.

- ***Last Time Modified***: Applications in Google Play go through changes and updates. The date of last modification is a metadata.
- ***Category***: Google Play categorizes applications based on their types, such as games, applications and book. Each game type further subcategorized as the action, adventure, arcade and board.
- ***Description***: Developers provide a brief description to describe the main functionality of the applications.
- ***Permissions***: Upon opting to install an application, it prompts the user with the list of permissions that the application requires to function properly.
- ***Rating***: Users rate every application based on their experience with the application. It is helpful for new users to decide whether to download the application.
- ***Creator ID***: Every developer has an ID in Google Play. They use their ID to publish the applications. In case of detecting a malicious application, Google is able to identify the developer and terminates the developers ID.

Discussions

In this paper, we looked back at the related works with respect to feature selection in Android malware detection. We categorized feature selection of Android malware detection into four groups (i.e. static, dynamic, hybrid and metadata). Such categorization assists researchers make decision on which features to choose. Additionally, they get to know selected features by other researchers. In this section, we discuss guidelines on feature selection. Moreover, we talk about available datasets of Android malware. We also review evaluation measures used in recent research works. Finally, we present current challenges and open research areas to conclude our discussions.

Feature selection guidelines

Choosing appropriate features is an important step in conducting an experiment that determines effectiveness and results of a research work. Android applications have many features. We suggest the following two approaches for feature selection based on reviewed papers.

- **Selection based on rationalizing**: As mentioned in the Section 3.1.1, permissions are static feature and the first line of defense in Android applications against the attacker, which is a plausible reason for choosing it as a feature. Among static features, research community has been paying a considerable attention to permissions of Android applications. This signifies that authors comprehended effectiveness of this feature and chose them based on reasoning.
  Java code is another static feature used widely in recent works. Java code is the source of malicious activities of malware and undoubtedly is a focus of research works. However, analyzing Java code is more difficult than analyzing permissions since attackers employ various techniques (e.g. obfuscation, encryption) to evade available detection methods (Petsas et al.). Thus, researchers have been developing complex methods to detect threats in Java code (Suarez-Tangil et al., 2014; Lu et al., 2012; Deshotels et al.). We believe that using Java code is more complex in malware detection than permissions and requires developing aggressive methods.
  Among dynamic features, Feizollah et al. (2013) used network traffic feature to detect mobile malware. The authors selected three network features, namely, the TCP size, connection duration and number of GET/POST parameters. They provided justification for using each of the parameters. Appropriate and justified selection of features led to 99.94% of detection rate. Another related work is (Shabtai et al., 2014) in which authors use network traffic for mobile malware detection.

However, researchers have used system call much more frequently than other dynamic features. Although all activities are done through system calls and selecting them as a feature is appropriate, we argue that network traffic of Android applications is a precious source for mobile malware analysis and only a few of the contemporary research works have focused on such features (refer to Section 3.2.2).

- **Selection based on feature ranking algorithms**: We found out that only 8 out of 100 reviewed papers employed feature ranking algorithms. Feature selection and ranking is task of selecting subset of original features that provides the most useful and important features based on the developed algorithms (Jensen and Shen, 2008). The algorithm receives collected dataset and uses mathematical calculations to rank features in the dataset. Information gain algorithm has been widely used for feature selection based on the entropy difference between the cases of using a feature against otherwise (Hyo-Sik and Mi-Jung, 2013). Shabtai and Elovici used feature ranking algorithms to select subset of features from 88 collected features. They managed to select top 10, 20 and 50 features from original dataset. Comparably, Shabtai et al. (2014) analyze network traffic of Android applications. They used feature selection algorithms to select the most useful features among massive features in network traffic data. Similarly, in (Shabtai et al., 2010a) the authors collected 2285 Android applications and extracted more than 9898 features. They then used feature selection algorithms to choose top 50, 100, 200, 300, 500 and 800 features.

Table 4 compares advantages and disadvantages of various features based on which researchers are able to make decisions.

Datasets

In this section, we want to emphasize on the importance of dataset in experiments. Every experiment requires a dataset based on which authors evaluate their proposed system. Android malware is a relatively new research area. The first Android malware was discovered in 2010 (Lookout, 2010). Initially, researchers did not have a solid and standard dataset of samples to work with. Instead, they tended to write their own malware and assessed their system on self-written malware. Other researchers tried to collect samples through some websites, which shared Android malware samples, such as Contagio (2014). Therefore, the weakness was limitation of malware samples that in turn made the evaluation of their system unreliable. In 2012, MalGenome (Yajin and Xuxian, 2012) data sample was released that contains 1260 malware samples categorized into 49 different malware families. It is a collection of malware from August 2010 to October 2011. The availability of such a valuable data sample filled the gap for most researchers. Based on our study, 24 out of 100 papers used MalGenome as their data sample. Table 5 shows the distribution of MalGenome usage in the related works. It is worth noting that the authors released MalGenome in 2012 and four research works used it in the same year. In 2013, the number raised to more than triple of 2012, which shows that researchers welcome a standard and solid data sample. As of reviewing related works, in March 2014, seven papers, (Yerima et al., 2014; F-Secure, 2014; Gianazza et al., 2014; Ham and Lee, 2014; Ham et al., 2014; Deshotels et al., 2556; Seo et al., 2014), used MalGenome, which is half of all 2013.

However, based on the nature of malware, they change shape and infecting technique to evade detection. Therefore, it behooves researchers to update the data samples to develop systems that are more effective. By introducing DREBIN (Arp et al., 2014) in 2014, such need was fulfilled. DREBIN is a collection of 5560 malware from 179 different families, which were collected between August 2010 and October 2012. In 2014, this data sample was used in 19 research works. Thus, it shows that research community is keen on using standardized data samples.

Evaluation measures

Researchers assess the effectiveness of a proposed system by how accurate it can detect malware using various evaluation measures. We analyzed the measures utilized in the literature. Table 6 shows some of the references along with their respective measures. It is vital to discuss common evaluation measures in the research community. Below is the definition of each one.

Confusion Matrix: Results of an experiment can be represented in the form of a table known as confusion matrix (Davis and Goadrich). It has four categories as following:

- **True Positive (TP)**: It is the number of correctly classified instances as positive. It means that how successful a system is in detecting a malware as malicious. As the true positive increases, the result is better.

Table 4
Comparison of different types of features.

	Advantages	Disadvantages
Static features	Easy to extract	Obfuscation
Dynamic features	More comprehensive than static features	Code coverage Difficult to extract Requires rooted device
Hybrid features	The most comprehensive collection of features	Extraction process is complex Have to extract static and dynamic features
Applications metadata	Easy to extract	Not widely used among researchers

Table 5
Use of MalGenome data sample in the reviewed works.

Year	Number of papers
2010	0
2011	0
2012	4
2013	13
2014 (First Quarter)	7

Table 6
Evaluation measures in selected reviewed papers.

Evaluation measures	Type of features	Reference
Confusion matrix	Static Features	Wu et al. (2012)
	Dynamic Features	Su et al. (2012)
	Applications Metadata	Pandita et al.
True positive	Static Features	Grace et al.
	Dynamic Features	Zhao et al.
	Dynamic Features	Iland et al. (2011)
	Hybrid Features	Kim et al. (2013)
Accuracy	Static Features	Wu et al. (2012)
	Static Features	Sanz et al. (2013a)
	Dynamic Features	Burguera et al.
	Applications Metadata	Pandita et al.
Precision	Static Features	Wu et al. (2012)
	Applications Metadata	Pandita et al.
True positive rate	Static Features	Peng et al. (2012)
	Static Features	Sanz et al. (2013a)
	Dynamic Features	Feizollah et al. (2013)
F-measure	Static Features	Wu et al. (2012)
	Applications Metadata	Pandita et al.
ROC	Static Features	Peng et al. (2012)
	Static Features	Sanz et al. (2013a)
	Dynamic Features	Feizollah et al. (2013)

- **False Positive (FP)**: It is the number of incorrectly classified instances as positive. It means that the ratio of which the algorithm considers normal data as malicious. As the false positive decreases, it shows that the system is more accurate.
- **True Negative (TN)**: It is the number of correctly classified instances as negative.
- **False Negative (FN)**: It is the number of incorrectly classified instances as negative.

Table 7 shows the confusion matrix as a table. It is worth noting that some researchers calculate true positive that is detection rate and false positive that is false alarm only, since they are more important measures than others.

- **Accuracy**: It shows how accurate the system can detect malware. As an example, accuracy of 0.6 implies that the system is capable of detecting 60 malware from dataset of 100 malware.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

- **Precision**: It is the number of instances correctly classified as class *X* among those classified as class *X*. Simply put, the precision addresses the following question. Based on prediction, how likely is it that the prediction be true?

Table 7
Confusion matrix.

	Actual positive	Actual negative
Predicted positive	TP	FP
Predicted negative	FN	TN

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

- **Recall**: It is equivalent to the true positive rate (TPR). It assumes that there are malware in dataset and asks this question. Is the algorithm going to detect malware? Therefore, it measures performance of the algorithm.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

- **F-measure**: It is weighted harmonic mean of precision and recall. Researchers seldom used this measure in research works.

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \quad (4)$$

- **ROC**: Receiver Operating Characteristic (ROC) curve indicates how a detection rate change as the internal threshold changes to generate more or fewer false alarm. It plots intrusion detection accuracy against false positive probability. Area Under the Curve (AUC) often accompanies the ROC curve. It is the area under the ROC curve. Its value varies between 0 and 1. As it approaches 1, the system has better performance.

Challenges and open research areas

Although there have been many research works in the mobile malware detection field, there are some challenges still available. Based on reviewed papers, we explain them as follows. Additionally, we mention open research areas that have potentials in researching.

Virtual environment

With the advent of mobile malware, researchers used virtual environment to evaluate malware behavior. As time passed by, attackers became aware of utilizing virtual environments. As a result, we are witnessing new types of mobile malware that are knowledgeable about virtual environment and they are able to detect if they are active in such environment. F-Secure published a report about Dendroid (F-Secure, 2014) in which it mentions the awareness of this malware about virtual environment. It hides its malicious behavior when it is in virtual environment. Similarly, Android.hehe appeared in 2014 that is capable of detecting virtual environment and emulators and hides its malicious behaviors (Dharmdasani, 2014). Research works were published recently that addressed the malware awareness about virtual environment (Petsas et al.; Vidas and Christin). There is a need in developing detection methods to discover such malware.

Obfuscation

One of the techniques in mobile malware detection is using the Java code and generating unique signatures based on classes, fields and methods in the code. Obfuscation is a method that attackers use in order to evade the detection process by renaming classes, fields and methods. As a result, the generated signature is different from what detection systems have. Some research works have addressed such issues and proposed methods against obfuscation such as (Christodorescu et al.; Fredrikson et al., 2010; Kolbitsch et al.). They used call graph that uses graph to draw the flow of system calls. However, we cannot implement such solutions on available tools due to low privileges of anti-malware tools on Android. Thus, there is a need to develop effective solutions for this existing challenge.

Code coverage

Researchers often argue that dynamic analysis of a malware results in exposing some of its malicious behavior, but not all of them. Code coverage is covering every possible execution of applications code. Sasnauskas and Regehr (2014) mention that producing highly structured inputs that get high code coverage is an open research challenge. AppsPlayground (Rastogi et al.) is a framework that analysis GUI and injects random events to trigger malicious behaviors. However, its code coverage is 33%. Other works such as (Bierma et al., 2014; Gilbert et al., 2014) mentioned code coverage as limitation of their work or as future research.

Wearable devices

Wearable devices are new generation of computing technology. It is estimated that the global market revenue of such devices to cross USD $8 billion in 2014 (marketsandmarkets, 2014). In addition, the wearable market is expected to grow by 350% in 2014, based on a report from CNBC (CNBC, 2014). The operating system of such devices is a modified version of Android, which raises concern over security issue. Google Glass is a famous wearable device developed by Google (Google, 2014d). Security analysts expressed their concern over Google Glass, since it is another Android platform and it is source of valuable information. The main input is the camera. The attacker could see everything the victim sees. This could include banking login information, two-factor authentication codes or possibly extorting money from a victim by capturing embarrassing video (Security_Watch, 2013). Thus, it is a new research area in security field, which is worth researching.

New standardized dataset

As discussed before, Android malware are changing rapidly. They have moved toward stealing data and hijacking mobile devices. Ransomware are on the rise that steal users data or encrypt mobile devices and they demand ransom. As an instance, researchers discovered a Trojan-ransom in May 2014. It uses standard HTTP communication and asks for money. When they ask for money from the user, they show users image using front camera of the device (Unuchek, 2014). Because of the aforementioned changes in malware, researchers need to

Table 8
List of all reviewed papers.

No.	Reference	Type of feature	Year	No. of tested apps
1	Zhemin and Min	Static	2012	1750
2	Arzt et al. (2014)	Static	2014	–
3	Yerima et al. (2014)	Static	2013	2000
4	Desnos (2012)	Static	2012	–
5	Apvrille and Apvrille (2013)	Static	2013	–
6	Aung and Zaw (2013)	Static	2013	500
7	Grace et al.	Static	2013	–
8	Feng et al. (2013)	Static	2013	–
9	Rastogi et al. (2014)	Static	2014	–
10	Faruki et al.	Static	2013	6779
11	Suarez-Tangil et al. (2014)	Static	2014	1231
12	Rosen et al.	Static	2013	2782
13	Peng et al. (2012)	Static	2012	325,036
14	Grace et al.	Static	2012	118,318
15	Lu et al. (2012)	Static	2012	5486
16	Crussell et al.	Static	2012	9400
17	Sarma et al.	Static	2012	158,062
18	Samra et al. (2013)	Static	2013	18,174
19	Arp et al. (2014)	Static	2014	129,013
20	Deshotels et al.	Static	2014	1100
21	Luoshi et al.	Static	2013	–
22	Gascon et al. (2013)	Static	2013	12,158
23	Sanz et al. (2013b)	Static	2013	3013
24	Walenstein et al. (2012)	Static	2012	–
25	Sanz et al.	Static	2013	666
26	Wu et al. (2012)	Static	2012	1738
27	Huang et al.	Static	2012	125,249
28	Zhou et al.	Static	2012	91,093
29	Aafer et al.	Static	2013	20,000
30	Lee and Jin (2013)	Static	2013	–
31	Yerima et al. (2013)	Static	2013	2000
32	Shabtai et al. (2010a)	Static	2010	2285
33	Sahs and Khan (2012)	Static	2012	2172
34	Zheng et al. (2013)	Static	2013	150,368
35	Sanz et al. (2013a)	Static	2013	2060
36	Zhou et al.	Static	2013	84,767
37	Huang et al.	Static	2014	182
38	Almohri et al.	Static	2014	405
39	Zheng et al.	Static	2014	24,009
40	Sanz et al.	Static	2014	2144
41	Paturi et al. (2013)	Static	2013	–
42	Seo et al. (2014)	Static	2014	1257
43	Rasthofer et al. (2014)	Static	2014	11,000
44	Liang et al.	Static	2013	52
45	Wu et al.	Static	2014	–
46	Tchakount and Dayang (2013)	Dynamic	2013	–
47	Hyo-Sik and Mi-Jung (2013)	Dynamic	2013	14,794
48	Yu et al. (2013)	Dynamic	2013	–
49	Shabtai and Elovici	Dynamic	2010	43
50	Chekina et al.	Dynamic	2012	10
51	Backes et al.	Dynamic	2012	–
52	Baliga et al. (2013)	Dynamic	2013	9
53	Rastogi et al.	Dynamic	2013	3968
54	Burguera et al.	Dynamic	2011	–
55	Yan and Yin	Dynamic	2012	–
56	Dini et al.	Dynamic	2012	56
57	Enck et al. (2014)	Dynamic	2010	30
58	Portokalidis et al.	Dynamic	2010	–
59	Choi et al.	Dynamic	2013	–
60	Gianazza et al. (2014)	Dynamic	2014	15
61	Ham and Lee (2014)	Dynamic	2014	1257
62	Ham et al. (2014)	Dynamic	2014	1257
63	Zhang et al. (2013)	Dynamic	2013	1249
64	Su et al. (2012)	Dynamic	2012	120
65	Maggi et al.	Dynamic	2013	18,758
66	Zhao et al.	Dynamic	2011	200
67	Shabtai et al. (2014)	Dynamic	2014	500,000
68	Xiaoming and Qiaoyan (2011)	Dynamic	2011	–

Table 8 (*continued*)

No.	Reference	Type of feature	Year	No. of tested apps
69	Houmansadr et al. (2011)	Dynamic	2011	–
70	Iland et al. (2011)	Dynamic	2011	18
71	Amos et al. (2013)	Dynamic	2013	1738
72	Karami et al. (2013)	Dynamic	2013	20
73	Damopoulos et al. (2012)	Dynamic	2011	–
74	Reina et al.	Dynamic	2013	1200
75	Khune and Thangakumar (2012)	Dynamic	2012	–
76	Zonouz et al.2013)	Dynamic	2013	–
77	Isohara et al. (2011)	Dynamic	2011	230
78	Feizollah et al. (2013)	Dynamic	2013	1257
79	Feizollah et al. (2014)	Dynamic	2014	1000
80	Hoffmann et al. (2013)	Dynamic	2013	–
81	Lu et al. (2014)	Dynamic	2014	331
82	Lin et al. (2013)	Dynamic	2013	100
83	Shabtai et al. (2010b)	Dynamic	2010	5
84	Veen (2013)	Dynamic	2013	–
85	Bente (2013)	Dynamic	2013	–
86	Machiry et al.	Dynamic	2013	50
87	Jang et al.	Dynamic	2014	709
88	Spreitzenbarth et al.	Hybrid	2013	36,000
89	Zhou et al.	Hybrid	2012	204,040
90	Moonsamy et al. (2013)	Hybrid	2013	1227
91	Wei et al.	Hybrid	2012	27
92	Eder et al. (2013)	Hybrid	2013	1260
93	Blasing et al. (2010)	Hybrid	2010	–
94	Kim et al. (2013)	Hybrid	2013	1003
95	Xu et al. (2013)	Hybrid	2013	100,000
96	Zheng et al.	Hybrid	2012	19
97	Shalaginov and Franke	Hybrid	2014	604
98	Guido et al. (2013)	Metadata	2013	–
99	Teufl et al. (2013)	Metadata	2013	–
100	Pandita et al.	Metadata	2013	–

evaluate new systems on new types of malware, rather than old ones. It benefits the research community to publish Android malware dataset similar to (Yajin and Xuxian, 2012; Arp et al., 2014) along with comprehensive analysis of the malware. So that researchers develop effective systems based on the analysis.

Conclusion

In this review paper, we analyzed 100 papers with the perspective of feature selection in Android malware detection. We categorized features of Android malware into four groups. The first group comprised of static features that are pertaining to Android installation file itself prior to installation on the device. The second group includes dynamic features that are pertaining to the behavior of the application after installation. The third group is hybrid features that are combination of both static and dynamic features. The last group was metadata that is any related data on Google Play. We examined each group in details. We also delved into discussing datasets used in the literature along with the evaluation measures utilized in recent works. Furthermore, we listed all the reviewed papers in Table 8 for researchers to have a glance view of recent works. It is worth mentioning that some papers introduced novel methods, however due to lack of malware sample, authors could not test their systems thoroughly (e.g. (Shabtai et al., 2010b)).

Acknowledgments

This work was supported in part by the Ministry of Higher Education, Malaysia, under Grant FRGS FP034-2012A and the Ministry of Science, Technology and Innovation, under Grant eScienceFund 01-01-03-SF0914.

References

Aafer, Y., Du, W., Yin, H., Droidapiminer: Mining api-level features for robust malware detection in android, In: 9th International Conference on Security and Privacy in Communication Networks, pp. 86–103. URL http://dx.doi.org/10.1007/978-3-319-04283-1_6.

Almohri, H. M., Yao, D., Kafura, D., Droidbarrier: know what is executing on your android, in: 4th ACM conference on Data and application security and privacy, pp. 257–264, (Daphne). http://dx.doi.org/10.1145/2557547.2557571.

Amos B, Turner H, White J. Applying machine learning classifiers to dynamic android malware detection at scale. In: 9th International Wireless Communications and Mobile Computing Conference (IWCMC); 2013. p. 1666–71. http://dx.doi.org/10.1109/IWCMC.2013.6583806.

Apple. Apple – ios 7. 2014. URL, www.apple.com/ios.

Aptoide. Aptoide — android apps store. 2014. URL, http://www.aptoide.com/.

Apvrille L, Apvrille A. Pre-filtering mobile malware with heuristic techniques. Report. 2013. URL, http://www.fortiguard.com/paper/Pre-filtering-Mobile-Malware-with-Heuristic-Techniques.

Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K. Drebin: effective and explainable detection of android malware in your pocket. In: Network and Distributed System Security (NDSS) Symposium; 2014.

Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, et al. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. ACM SIGPLAN Not 2014;49(6):259–69. http://dx.doi.org/10.1145/2666356.2594299.

Aung Z, Zaw W. Permission-based android malware detection. Int J Sci Technol Res 2013;2(3):228–34.

AVG.ThreatLabs. Android/dowgin. 2013. URL, http://www.avgthreatlabs.com/virus-and-malware-information/info/android-dowgin/.

Backes, M., Gerling, S., Hammer, C., Maffei, M., Styp-Rekowsky, P. v., Appguard: enforcing user requirements on android apps, in: 19th international conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 543–548. http://dx.doi.org/10.1007/978-3-642-36742-7_39.

Baeza-Yates RA, Ribeiro-Neto. Modern information retrieval. Boston: Addison-Wesley Longman Publishing Co.; 1999.

Baliga A, Bickford J, Daswani N. Titan: a carrier-based approach for detecting and mitigating mobile malware. Report, AT&T. 2013. URL, http://www.research.att.com/techdocs/TD_101245.pdf.

Bente I. Towards a network-based approach for smartphone security. Thesis. 2013. URL, https://137.193.200.7/doc/90756/90756.pdf.

Bierma M, Gustafson E, Erickson J, Fritz D, Choe YR. Andlantis: large-scale android dynamic analysis. Report. 2014. URL, http://mostconf.org/2014/papers/s3p2.pdf.

Blasing T, Batyuk L, Schmidt A-D, Camtepe SA, Albayrak S. An android application sandbox system for suspicious software detection. In: 5th International Conference on Malicious and Unwanted Software (MALWARE); 2010. p. 55–62. http://dx.doi.org/10.1109/MALWARE.2010.5665792.

Burguera, I., Zurutuza, U., Nadjm-Tehrani, S., Crowdroid: behavior-based malware detection system for android, in: 1st ACM workshop on security and privacy in smartphones and mobile devices, pp. 15–26. http://dx.doi.org/10.1145/2046614.2046619.

Chekina, L., Mimran, D., Rokach, L., Elovici, Y., Shapira, B., Detection of deviations in mobile applications network behavior. URL, http://arxiv.org/abs/1208.0564.

Chen BX, Bilton N. Building a better battery. 2014. URL, http://www.nytimes.com/2014/02/03/technology/building-a-better-battery.html?_r=0.

Chin, E., Felt, A. P., Greenwood, K., Wagner, D., Analyzing inter-application communication in android, In: 9th international conference on Mobile systems, applications, and services, pp. 239–252. http://dx.doi.org/10.1145/1999995.2000018.

Choi, S., Sun, K., Eom, H., Android malware detection using library api call tracing and semantic-preserving signal processing techniques, Report.

Christodorescu, M., Jha, S., Kruegel, C., Mining specifications of malicious behavior, In: 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 1287628, pp. 5–14. http://dx.doi.org/10.1145/1287624.1287628.

CNBC. Wearable smart bands set for 350growth in 2014. 2014. URL, http://www.cnbc.com/id/101410507.

CNET. Android dominates 81 percent of world smartphone market. 2013. URL, http://news.cnet.com/8301-1035_3-57612057-94/android-dominates-81-percent-of-world-smartphone-market/.

Contagio. Contagio. 2014. URL, http://contagiodump.blogspot.com/.

Crussell, J., Gibler, C., Chen, H., Attack of the clones: Detecting cloned applications on android markets, In: 17th European Symposium on Research in Computer Security, Lecture Notes in Computer Science, pp. 37–54. URL http://dx.doi.org/10.1007/978-3-642-33167-1_3.

Crussell, J., Gibler, C., Chen, H., Attack of the clones: Detecting cloned applications on android markets, In: 17th European Symposium on Research in Computer Security, Lecture Notes in Computer Science, pp. 37–54. URL http://dx.doi.org/10.1007/978-3-642-33167-1_3.

Damopoulos D, Menesidou SA, Kambourakis G, Papadaki M, Clarke N, Gritzalis S. Evaluation of anomaly-based ids for mobile devices using machine learning classifiers. Secur Commun Netw 2012;5(1):3–14. URL, http://dx.doi.org/10.1002/sec.341.

Davis, J., Goadrich, M., The relationship between precision-recall and roc curves, In: 23rd international conference on Machine learning, pp. 233–240. http://dx.doi.org/10.1145/1143844.1143874.

Deshotels, L., Notani, V., Lakhotia, A., Droidlegacy: Automated familial classification of android malware, In: ACM SIGPLAN on Program Protection and Reverse Engineering Workshop, pp. 1–12. http://dx.doi.org/10.1145/2556464.2556467.

Desnos A. Android: static analysis using similarity distance. In: 45th Hawaii International Conference on System Science (HICSS); 2012. p. 5394–403. http://dx.doi.org/10.1109/HICSS.2012.114.

Dharmdasani H. Android.hehe: malware now disconnects phone calls. 2014. URL, https://www.fireeye.com/blog/threat-research/2014/01/android-hehe-malware-now-disconnects-phone-calls.html.

Dini, G., Martinelli, F., Saracino, A., Sgandurra, D., Madam: A multi-level anomaly detector for android malware, in: 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, Lecture Notes in Computer Science, pp. 240–253. URL http://dx.doi.org/10.1007/978-3-642-33704-8_21.

Eder T, Rodler M, Vymazal D, Zeilinger M. Ananas – a framework for analyzing android applications. In: Eighth International Conference on Availability, Reliability and Security (ARES); 2013. p. 711–9. http://dx.doi.org/10.1109/ARES.2013.93.

Enck W, Gilbert P, Chun B-G, Cox LP, Jung J, McDaniel P, et al. Taintdroid: an information flow tracking system for real-time privacy monitoring on smartphones. Commun ACM 2014;57(3):99–106. http://dx.doi.org/10.1145/2494522.

Eset. Eset virusradar. 2013. URL, http://www.virusradar.com/en/Android_Adware.Dowgin/chart/history.

M. Eslahi, R. Salleh, N. B. Anuar, Mobots: a new generation of botnets on mobile devices and networks, in: 2012 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), IEEE, pp. 262–266. doi:http://dx.doi.org/10.1109/ISCAIE.2012.6482109.

F-Secure. Backdoor:android/dendroid.a. 2014. URL, http://www.f-secure.com/v-descs/backdoor_android_dendroid_a.shtml.

Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., Bharmal, A., Androsimilar: robust statistical feature signature for android malware detection, In: 6th International Conference on Security of Information and Networks, pp. 152–159. http://dx.doi.org/10.1145/2523514.2523539.

Feizollah A, Anuar NB, Salleh R, Amalina F, Maarof RR, Shamshirband S. A study of machine learning classifiers for anomaly-based mobile botnet detection. Malays J Comput Sci 2013;26(4):251–65.

Feizollah A, Anuar NB, Salleh R, Amalina F. Comparative study of k-means and mini batch kmeans clustering algorithms in android malware detection using network traffic analysis. In: International Symposium on Biometrics and Security Technologies (ISBAST); 2014. p. 198–202.

Felt, A. P., Chin, E., Hanna, S., Song, D., Wagner, D., Android permissions demystified, in: 18th ACM conference on Computer and communications security, pp. 627–638. http://dx.doi.org/10.1145/2046707.2046779.

Feng Y, Anand S, Dillig I, Aiken A. Apposcopy: semantics-based detection of android malware. Report. 2013. URL, www.cs.utexas.edu/~isil/fse14.pdf.

Fortinet. Fortinet's fortiguard labs reports 96.5% of all mobile malware tracked is android-based. 2014. URL, http://www.fortinet.com/resource_center/whitepapers/threat-landscape-report-2014.html.

Fredrikson M, Jha S, Christodorescu M, Sailer R, Yan X. Synthesizing near-optimal malware specifications from suspicious behaviors. In: IEEE Symposium on Security and Privacy; 2010. p. 45–60. http://dx.doi.org/10.1109/sp.2010.11.

Garcia-Teodoro P, Diaz-Verdejo J, Maci-Fernndez G, Vzquez E. Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur 2009;28(1):18–28. URL, http://dx.doi.org/10.1016/j.cose.2008.08.003.

Gartner. Gartner says worldwide pc, tablet and mobile phone shipments to grow 5.9 percent in 2013 as anytime-anywhere-computing drives buyer behavior. 2013. URL, http://www.gartner.com/newsroom/id/2525515.

Gascon H, Yamaguchi F, Arp D, Rieck K. Structural detection of android malware using embedded call graphs. In: ACM workshop on Artificial intelligence and security; 2013. p. 45–54. http://dx.doi.org/10.1145/2517312.2517315.

Gianazza A, Maggi F, Fattori A, Cavallaro L, Zanero S. Puppetdroid: a user-centric ui exerciser for automatic dynamic analysis of similar android applications. 1st April 2014. 2014. URL, http://arxiv.org/abs/1402.4826.

Gilbert C, Cronkite-Ratcliff B, Franklin J. Malbehave: classifying malware by observed behavior. Report. 2014. URL, https://www.connorgilbert.com/.

Google. Android. 2014. URL, www.android.com.

Google. permission. 2014. URL, http://developer.android.com/guide/topics/manifest/permission-element.html.

Google. Introducing art. 2014. URL, https://source.android.com/devices/tech/dalvik/art.html.

Google. Google glass. 2014. URL, http://www.google.com/glass/start/.

M. Grace, Y. Zhou, Q. Zhang, S. Zou, X. Jiang, Riskranker: scalable and accurate zero-day android malware detection, in: 10th international conference on Mobile systems, applications, and services, pp. 281–294. doi:10.1145/2307636.2307663.

Grace, M., Zhou, Y., Wang, Z., Jiang, X., Systematic detection of capability leaks in stock android smartphones, In: 19th Network and Distributed System Security Symposium. URL, http://www4.ncsu.edu/~zwang15/files/NDSS12_Woodpecker.pdf.

Guido M, Ondricek J, Grover J, Wilburn D, Nguyen T, Hunt A. Automated identification of installed malicious android applications. Digit Investig 2013;10(Suppl. (0)):S96–104. URL, http://www.sciencedirect.com/science/article/pii/S1742287613000571.

Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res 2003;3:1157–82.

Ham YJ, Lee H-W. Detection of malicious android mobile applications based on aggregated system call events. Int J Comput Commun Eng 2014;3(2):149–54.

Ham YJ, Moon D, Lee H-W, Lim JD, Kim JN. Android mobile application system call event pattern analysis for determination of malicious attack. Int J Secur Appl 2014;8(1). 241–236.

Hoffmann, J., Neumann, S., Holz, T., Mobile malware detection based on energy fingerprints a dead end?, In: 16th International Symposium, RAID 2013, Lecture Notes in Computer Science, pp. 348–368. URL http://dx.doi.org/10.1007/978-3-642-41284-4_18.

Houmansadr, A., Zonouz, S. A., Berthier, R., A cloud-based intrusion detection and response system for mobile phones, In: 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 31–32. http://dx.doi.org/10.1109/DSNW.2011.5958860.

Huang, C.-Y., Tsai, Y.-T., Hsu, C.-H., Performance evaluation on permission-based detection for android malware, In: International Computer Symposium ICS, Smart Innovation, Systems and Technologies, pp. 111–120. URL http://dx.doi.org/10.1007/978-3-642-35473-1_12.

Huang, J., Zhang, X., Tan, L., Wang, P., Liang, B., Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction, In: 36th International Conference on Software Engineering, 1036–1046. URL, http://doi.acm.org/10.1145/2568225.2568301.

Hyo-Sik H, Mi-Jung C. Analysis of android malware detection performance using machine learning classifiers. In: International Conference on ICT Convergence (ICTC); 2013. p. 490–5. http://dx.doi.org/10.1109/ICTC.2013.6675404.

Iland D, Pucher A, Schauble T. Detecting android malware on network level. Report. UC Santa Barbara; 2011. URL, http://cs.ucsb.edu/~iland/AndroidMalwareDetection.pdf.

Isohara, T., Takemori, K., Kubota, A., Kernel-based behavior analysis for android malware detection, in: 2011 Seventh International Conference on Computational Intelligence and Security (CIS), pp. 1011–1015. http://dx.doi.org/10.1109/CIS.2011.226.

Jang, J.-W., Yun, J., Woo, J., Kim, H. K., Andro-profiler: anti-malware system based on behavior profiling of mobile malware, In: companion

publication of the 23rd international conference on World wide web companion, pp. 737–738. http://dx.doi.org/10.1145/2567948.2579366.
Jensen R, Shen Q. Computational intelligence and feature selection: rough and fuzzy approaches. New Jersey, USA: John Wiley & Sons; 2008.
Karami M, Elsabagh M, Najafiborazjani P, Stavrou A. Behavioral analysis of android applications using automated instrumentation. In: IEEE 7th International Conference on Software Security and Reliability-Companion (SERE-C); 2013. p. 182–7. http://dx.doi.org/10.1109/SERE-C.2013.35.
Khune RS, Thangakumar J. A cloud-based intrusion detection system for android smartphones. In: International Conference on Radar, Communication and Computing (ICRCC); 2012. p. 180–4. http://dx.doi.org/10.1109/ICRCC.2012.6450572.
Kim, D.-U., Kim, J., Kim, S., A malicious application detection framework using automatic feature extraction tool on android market, in: 3rd International Conference on Computer Science and Information Technology (ICCSIT2013), pp. 4–5.
Knoernschild K. Rich mobile application platforms for the smartphone 2010. Report. Burton Group; 2010. URL, http://blogs.stlawu.edu/mobile/files/2010/07/rma2010.pdf.
Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, E., Zhou, X., Wang, X., Effective and efficient malware detection at the end host, In: 18th conference on USENIX security symposium, pp. 351–366.
La Polla M, Martinelli F, Sgandurra D. A survey on security for mobile devices. IEEE Commun Surv Tutor 2013;15(1):446–71. http://dx.doi.org/10.1109/SURV.2012.013012.00028.
Lee S-H, Jin S-H. Warning system for detecting malicious applications on android system. Int J Comput Commun Eng 2013;2(3):324–7.
Liang, S., Keep, A. W., Might, M., Lyde, S., Gilray, T., Aldous, P., Horn, D. V., Sound and precise malware analysis for android via pushdown reachability and entry-point saturation, In: Third ACM workshop on Security and privacy in smartphones mobile devices, pp. 21–32. http://dx.doi.org/10.1145/2516760.2516769.
Lin Y-D, Lai Y-C, Chen C-H, Tsai H-C. Identifying android malicious repackaged applications by thread-grained system call sequences. Comput Secur November 2013;39(Part B):340–50. URL, http://www.sciencedirect.com/science/article/pii/S0167404813001272.
Lookout. Security alert: Geinimi, sophisticated new android trojan found in wild. 2010. URL, https://blog.lookout.com/blog/2010/12/29/geinimi_trojan/.
Lookout. Mouabad.p: Pocket dialing for profit. 2013. URL, https://blog.lookout.com/blog/2013/12/09/mouabad-p-pocket-dialing-for-profit/.
Lu L, Li Z, Wu Z, Lee W, Jiang G. Chex: statically vetting android apps for component hijacking vulnerabilities. In: ACM conference on Computer and communications security; 2012. p. 229–40. http://dx.doi.org/10.1145/2382196.2382223.
Lu H, Zhao B, Su J, Xie P. Generating lightweight behavioral signature for malware detection in people-centric sensing. Wirel Personal Commun 2014;75(3):1591–609. URL, http://dx.doi.org/10.1007/s11277-013-1400-9.
Luoshi, Z., Yan, N., Xiao, W., Zhaoguo, W., Yibo, X., A3: Automatic analysis of android malware, In: 1st International Workshop on Cloud Computing and Information Security, pp. 89–93. http://www.atlantis-press.com/php/paper-details.php?id=9880.
Machiry, A., Tahiliani, R., Naik, M., Dynodroid: an input generation system for android apps, in: 9th Joint Meeting on Foundations of Software Engineering, pp. 224–234. http://dx.doi.org/10.1145/2491411.2491450.
Maggi, F., Valdi, A., Zanero, S., Andrototal: a flexible, scalable toolbox and service for testing mobile malware detectors, In: Third ACM workshop on Security and privacy in smartphones & mobile devices, pp. 49–54. http://dx.doi.org/10.1145/2516760.2516768.
marketsandmarkets. Wearable electronics market worth $8.36 billion by 2018. 2014. URL, http://www.marketsandmarkets.com/PressReleases/wearable-electronics.asp.
Microsoft. The smartphone reinvented around you. 2014. URL, http://www.windowsphone.com/en-us.
Mohite S, Sonar RS. A survey on mobile malware: war without end. Int J Comput Sci Bus Inf 2014;9(1):23–35.
Moonsamy V, Rong J, Liu S. Mining permission patterns for contrasting clean and malicious android applications. 2013. URL, http://www.sciencedirect.com/science/article/pii/S0167739X13001933.
Motley. Android dominates, blackberry nears the end. 2014. URL, http://www.fool.com/investing/general/2014/02/19/android-dominates-blackberry-nears-the-end.aspx.
Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T., Whyper: towards automating risk assessment of mobile applications, In: 22nd USENIX Security Symposium, pp. 527–542. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/pandita.
Paturi A, Cherukuri M, Donahue J, Mukkamala S. Mobile malware visual analytics and similarities of attack toolkits (malware gene analysis). In: International Conference on Collaboration Technologies and Systems (CTS); 2013. p. 149–54. http://dx.doi.org/10.1109/CTS.2013.6567221.
Peng H, Gates C, Sarma B, Li N, Qi Y, Potharaju R, et al. Using probabilistic generative models for ranking risks of android apps. In: ACM conference on Computer and communications security, ACM; 2012. p. 241–52. http://dx.doi.org/10.1145/2382196.2382224.
Peng S, Yu S, Yang A. Smartphone malware and its propagation modeling: a survey. IEEE Commun Surv Tutor 2014;16(2):925–41. http://dx.doi.org/10.1109/SURV.2013.070813.00214.
Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., Ioannidis, S., Rage against the virtual machine: hindering dynamic analysis of android malware, In: Seventh European Workshop on System Security, pp. 1–6. http://dx.doi.org/10.1145/2592791.2592796.
Portokalidis, G., Homburg, P., Anagnostakis, K., Bos, H., Paranoid android: versatile protection for smartphones, in: 26th Annual Computer Security Applications Conference, pp. 347–356. http://dx.doi.org/10.1145/1920261.1920313.
R. in Motion, Blackberry smartphones. 2014. URL, http://us.blackberry.com/.
Rasthofer S, Arzt S, Bodden E. A machine-learning approach for classifying and categorizing android sources and sinks. In: Network and Distributed System Security (NDSS) Symposium; 2014.
Rastogi V, Yan C, Xuxian J. Catch me if you can: evaluating android anti-malware against transformation attacks. IEEE Trans Inf Forensics Secur 2014;9(1):99–108. http://dx.doi.org/10.1109/TIFS.2013.2290431.
Rastogi, V., Chen, Y., Enck, W., Appsplayground: automatic security analysis of smartphone applications, in: third ACM conference on Data and application security and privacy, pp. 209–220. http://dx.doi.org/10.1145/2435349.2435379.
Reina, A., Fattori, A., Cavallaro, L., A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors, In: 6th European Workshop on Systems Security. http://security.di.unimi.it/~joystick/pubs/eurosec13.pdf.
Rosen, S., Qian, Z., Mao, Z. M., Appprofiler: a flexible method of exposing privacy-related behavior in android applications to end users, In: third ACM conference on Data and application security and privacy, pp. 221–232. http://dx.doi.org/10.1145/2435349.2435380.
Sahs J, Khan L. A machine learning approach to android malware detection. In: European Intelligence and Security Informatics Conference (EISIC). IEEE; 2012. p. 141–7. URL, http://dx.doi.org/10.1109/EISIC.2012.34.
Samra AAA, Yim K, Ghanem OA. Analysis of clustering technique in android malware detection. In: Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS); 2013. p. 729–33.
Sands W. Walker sands mobile traffic report q3 2013. 2013. URL, http://www.walkersandsdigital.com/Walker-Sands-Mobile-Traffic-Report-Q3-2013.
Sanz, B., Santos, I., Ugarte-Pedrero, X., Laorden, C., Nieves, J., Bringas, P., Anomaly detection using string analysis for android malware detection, In: International Joint Conference SOCO13-CISIS13-ICEUTE13, Advances in Intelligent Systems and Computing, pp. 469–478. URL http://dx.doi.org/10.1007/978-3-319-01854-6_48.
B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, On the automatic categorisation of android applications, in: 2012 IEEE Consumer Communications and Networking Conference (CCNC), pp. 149–153. http://dx.doi.org/10.1109/CCNC.2012.6181075.
Sanz B, Santos I, Laorden C, Ugarte-Pedrero X, Bringas P, Ivarez G. PUMA: permission usage to detect malware in android. book section 30. Advances in intelligent systems and computing, Vol. 189. Berlin Heidelberg: Springer; 2013. p. 289–98. URL, http://dx.doi.org/10.1007/978-3-642-33018-6_30.
Sanz B, Santos I, Laorden C, Ugarte-Pedrero X, Nieves J, Bringas PG, et al. Mama: manifest analysis for malware detection in android. Cybern Syst 2013;44(6–7):469–88. URL, http://dx.doi.org/10.1080/01969722.2013.803889.
Sanz, B., Santos, I., Ugarte-Pedrero, X., Laorden, C., Nieves, J., Bringas, P. G., Instance-based anomaly method for android malware detection, In: 10th International Conference on Security and Cryptography, pp. 387–394. http://paginaspersonales.deusto.es/isantos/publications/2013/sanz_2013_Instance.pdf.
Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I., Android permissions: a perspective combining risks and benefits, In:

17th ACM symposium on Access Control Models and Technologies, pp. 13–22. http://dx.doi.org/10.1145/2295136.2295141.
Sasnauskas R, Regehr J. Intent fuzzer: crafting intents of death. 2014. http://dx.doi.org/10.1145/2632168.2632169.
Security_Watch. Google glass malware: It's coming. 2013. URL, http://securitywatch.pcmag.com/mobile-security/313703-google-glass-malware-it-s-coming.
Seo S-H, Gupta A, Mohamed Sallam A, Bertino E, Yim K. Detecting mobile malware threats to homeland security through static analysis. J Netw Comput Appl 2014;38:43–53. URL, http://www.sciencedirect.com/science/article/pii/S1084804513001227.
Shabtai, A., Elovici, Y., Applying behavioral detection on android-based devices, In: Mobile Wireless Middleware, Operating Systems, and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 235–249. URL http://dx.doi.org/10.1007/978-3-642-17758-3_17.
Shabtai A, Fledel Y, Elovici Y. Automated static code analysis for classifying android applications using machine learning. In: International Conference on Computational Intelligence and Security (CIS); 2010. p. 329–33. http://dx.doi.org/10.1109/CIS.2010.77.
Shabtai A, Kanonov U, Elovici Y. Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method. J Syst Softw 2010;83(8):1524–37. URL, http://www.sciencedirect.com/science/article/pii/S0164121210000762.
Shabtai A, Tenenboim-Chekina L, Mimran D, Rokach L, Shapira B, Elovici Y. Mobile malware detection through analysis of deviations in application network behavior. Comput Secur June 2014;43:1–18. URL, http://www.sciencedirect.com/science/article/pii/S0167404814000285.
Shalaginov A, Franke K. Automatic rule-mining for malware detection employing neuro-fuzzy approach. Report. Norwegian Information Security Laboratory; 2014. URL, http://tapironline.no/fil/vis/1421.
Sophos. Angry birds malware – firm fined 50,000 for profiting from fake android apps. 2012. URL, http://nakedsecurity.sophos.com/2012/05/24/angry-birds-malware-fine/.
Sophos. Security threat report 2013. Report. 2013. URL, http://www.sophos.com/en-us/medialibrary/PDFs/other/sophossecuritythreatreport2013.pdf.
Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., Hoffmann, J., Mobile-sandbox: having a deeper look into android applications, in: 28th Annual ACM Symposium on Applied Computing, pp. 1808–1815. http://dx.doi.org/10.1145/2480362.2480701.
Su, X., Chuah, M., and Tan, G., Smartphone dual defense protection framework: Detecting malicious applications in android markets, in: 2012 Eighth International Conference on Mobile Ad-hoc and Sensor Networks (MSN), pp. 153–160. http://dx.doi.org/10.1109/MSN.2012.43.
Suarez-Tangil G, Tapiador J, Peris-Lopez P, Ribagorda A. Evolution, detection and analysis of malware for smart devices. IEEE Commun Surv Tutor 2013;PP(99):1–27. http://dx.doi.org/10.1109/SURV.2013.101613.00077.
Suarez-Tangil G, Tapiador JE, Peris-Lopez P, Blasco J. Dendroid: a text mining approach to analyzing and classifying code structures in android malware families. Expert Syst Appl 2014;41(4, Part 1):1104–17. URL, http://www.sciencedirect.com/science/article/pii/S0957417413006088.
Symantec. Android madware and malware trends. 2013. URL, http://www.symantec.com/connect/blogs/android-madware-and-malware-trends.
Symantec. The future of mobile malware. 2014. URL, http://www.symantec.com/connect/blogs/future-mobile-malware.
Tchakount F, Dayang P. System calls analysis of malwares on android. Int J Sci Technol 2013;2(9):669–74.
Techcrunch. Android Accounted For 79% Of All Mobile Malware In 2012, 96% In Q4 Alone. 2013. URL, http://techcrunch.com/2013/03/07/f-secure-android-accounted-for-79-of-all-mobile-malware-in-2012-96-in-q4-alone/.
Teufl P, Ferk M, Fitzek A, Hein D, Kraxberger S, Orthacker C. Malware detection by applying knowledge discovery processes to application metadata on the android market (Google play). 1st April 2014, URL, http://dx.doi.org/10.1002/sec.675.
The.Register. Earn 8,000 a month with bogus apps from Russian malware factories. 2013. URL, http://www.theregister.co.uk/2013/08/05/mobile_malware_lookout/.
Unuchek R. The first mobile encryptor trojan. 2014. URL, http://securelist.com/blog/mobile/63767/the-first-mobile-encryptor-trojan/.
Veen V v d. Dynamic analysis of android malware. Thesis. 2013. URL, http://tracedroid.few.vu.nl/thesis.pdf.
Vidas, T., Christin, N., Evading android runtime analysis via sandbox detection, In: 9th ACM symposium on Information, computer and communications security, pp. 447–458. http://dx.doi.org/10.1145/2590296.2590325.
Virustotal. Antivirus. 2013. URL, https://www.virustotal.com/en/file/3684a199b0dd9504fe331015f7f23a24414773c43c89e6112f9dd2f2f00bc053/analysis/.
Vural, I., Venter, H., Mobile botnet detection using network forensics, In: Third Future Internet Symposium, vol. 6369 of Lecture notes in computer science, Springer Berlin Heidelberg, pp. 57–67. URL http://dx.doi.org/10.1007/978-3-642-15877-3_7.
Walenstein, A., Deshotels, L., Lakhotia, A., Program structure-based feature selection for android malware analysis, In: 4th International Conference, MobiSec 2012, Vol. 107 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 51–52. URL http://dx.doi.org/10.1007/978-3-642-33392-7_5.
Wang, Y., Zheng, J., Sun, C., Mukkamala, S., Quantitative security risk assessment of android permissions and applications, In: 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Vol. 7964 of Lecture Notes in Computer Science, pp. 226–241. URL http://dx.doi.org/10.1007/978-3-642-39256-6_15.
Wei, X., Gomez, L., Neamtiu, I., Faloutsos, M., Profiledroid: multi-layer profiling of android applications, In: 18th annual international conference on Mobile computing and networking, pp. 137–148. http://dx.doi.org/10.1145/2348543.2348563.
Wu D-J, Mao C-H, Wei T-E, Lee H-M, Wu K-P. Droidmat: android malware detection through manifest and api calls tracing. In: Seventh Asia Joint Conference on Information Security (Asia JCIS). IEEE; 2012. p. 62–9. URL, http://dx.doi.org/10.1109/AsiaJCIS.2012.18.
Wu, C., Zhou, Y., Patel, K., Liang, Z., Jiang, X., Airbag: Boosting smartphone resistance to malware infection, in: 21th Annual Network and Distributed System Security Symposium (NDSS'14). URL, www.yajin.org/papers/ndss14_airbag.pdf.
Xiaoming, K., Qiaoyan, W., Intrusion detection model based on android, In: 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), pp. 624–628. http://dx.doi.org/10.1109/ICBNMT.2011.6156010.
Xu J, Yu Y, Chen Z, Cao B, Dong W, Guo Y, et al. Mobsafe: cloud computing based forensic analysis for massive mobile applications using data mining. Tsinghua Sci Technol 2013;18(4):418–27. http://dx.doi.org/10.1109/TST.2013.6574680.
Yajin Z, Xuxian J. Dissecting android malware: characterization and evolution. In: IEEE Symposium on Security and Privacy (SP); 2012. p. 95–109. http://dx.doi.org/10.1109/SP.2012.16.
Yan, L. K., Yin, H., Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis, In: 21st USENIX conference on Security symposium, pp. 29–29.
Yerima SY, Sezer S, McWilliams G, Muttik I. A new android malware detection approach using bayesian classification. In: IEEE 27th International Conference on Advanced Information Networking and Applications (AINA); 2013. p. 121–8. http://dx.doi.org/10.1109/AINA.2013.88.
Yerima SY, Sezer S, McWilliams G. Analysis of bayesian classification-based approaches for android malware detection. IET Inf Secur 2014;8(1):25–36.
Yu W, Chen Z, Xu G, Wei S, Ekedebe N. A threat monitoring system for smart mobiles in enterprise networks. 2013. http://dx.doi.org/10.1145/2513228.2513266.
Zhang Y, Yang M, Xu B, Yang Z, Gu G, Ning P, et al. Vetting undesirable behaviors in android apps with permission use analysis. In: ACM SIGSAC conference on Computer & communications security; 2013. p. 611–22. http://dx.doi.org/10.1145/2508859.2516689.
Zhao, M., Ge, F., Zhang, T., Yuan, Z., Antimaldroid: An efficient svm-based malware detection framework for android, In: Second International Conference ICICA, Communications in Computer and Information Science, pp. 158–166. URL http://dx.doi.org/10.1007/978-3-642-27503-6_22.
Zhemin, Y., Min, Y., Leakminer: Detect information leakage on android with static taint analysis, in: Third World Congress on Software Engineering (WCSE), pp. 101–104. http://dx.doi.org/10.1109/WCSE.2012.26.
Zheng M, Sun M, Lui J. Droidanalytics: a signature based analytic system to collect, extract, analyze and associate android malware. 2013. URL, http://arxiv.org/abs/1302.7212.
Zheng, C., Zhu, S., Dai, S., Gu, G., Gong, X., Han, X., Zou, W., Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications, In: second ACM workshop on Security and privacy in smartphones and mobile devices, pp. 93–104. http://dx.doi.org/10.1145/2381934.2381950.
Zheng, M., Sun, M., Lui, J. C., Droidray: a security evaluation system for customized android firmwares, in: 9th ACM symposium on

Information, computer and communications security, pp. 471–482. http://dx.doi.org/10.1145/2590296.2590313.

Zhou, W., Zhou, Y., Jiang, X., Ning, P., Detecting repackaged smartphone applications in third-party android marketplaces, In: second ACM conference on Data and Application Security and Privacy, pp. 317–326. http://dx.doi.org/10.1145/2133601.2133640.

Zhou, W., Zhou, Y., Grace, M., Jiang, X., Zou, S., Fast, scalable detection of "piggybacked" mobile applications, In: Third ACM conference on Data and application security and privacy, pp. 185–196. http://dx.doi.org/10.1145/2435349.2435377.

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: 19th Annual Network and Distributed System Security Symposium (NDSS), pp. 5–8.

Zonouz S, Houmansadr A, Berthier R, Borisov N, Sanders W. Secloud: a cloud-based comprehensive and lightweight security solution for smartphones. Comput Secur September 2013;37:215–27. URL, http://www.sciencedirect.com/science/article/pii/S016740481300031X.