Soft Comput (2016) 20:343–357
DOI 10.1007/s00500-014-1511-6

METHODOLOGIES AND APPLICATION

Evaluation of machine learning classifiers for mobile malware detection

Fairuz Amalina Narudin · Ali Feizollah · Nor Badrul Anuar · Abdullah Gani

Published online: 9 November 2014

Abstract Mobile devices have become a significant part of people's lives, leading to an increasing number of users involved with such technology. The rising number of users invites hackers to generate malicious applications. Besides, the security of sensitive data available on mobile devices is taken lightly. Relying on currently developed approaches is not sufficient, given that intelligent malware keeps modifying rapidly and as a result becomes more difficult to detect. In this paper, we propose an alternative solution to evaluating malware detection using the anomaly-based approach with machine learning classifiers. Among the various network traffic features, the four categories selected are basic information, content based, time based and connection based. The evaluation utilizes two datasets: public (i.e. MalGenome) and private (i.e. self-collected). Based on the evaluation results, both the Bayes network and random forest classifiers produced more accurate readings, with a 99.97 % true-positive rate (TPR) as opposed to the multi-layer perceptron with only 93.03 % on the MalGenome dataset. However, this experiment revealed that the k-nearest neighbor classifier efficiently detected the latest Android malware with an 84.57 % true-positive rate higher than other classifiers.

Communicated by V. Loia.

F. A. Narudin · A. Gani
Mobile Cloud Computing (MCC), University of Malaya, 50603 Kuala Lumpur, Malaysia

A. Feizollah (✉) · N. B. Anuar
Security Research Group (SECReg), Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
e-mail: ali.feizollah@siswa.um.edu.my

Keywords Intrusion detection system · Machine learning · Android malware detection · Anomaly based · Mobile device

1 Introduction

The tremendous growth in mobile and smart device usage has led to an increasing number of users connecting to the Internet. A survey conducted by TNS Infratest GmbH for Google shows that mobile device ownership is on the rise each month. For instance, in Japan, the number went up from 6 to 17 % in 2011 (Survey 2013). This technology yields many conveniences to end users, allowing communication with ease and no limitations. Today, activities like social gatherings, playing games and watching videos are feasible anywhere and anytime with mobile devices. The escalating number of users, however, also provides hackers with the opportunity to develop various malicious applications. In 2013, Symantec reported a significant 58 % ascent in malware families compared to 2011 and the Android platform was the most commonly targeted mobile device platform in 2012 (García-Teodoro et al. 2009). Similarly, Symantec published a report in September 2013 regarding the evolving ransomware throughout the Android operating system (Symantec 2013). Compared to all the different operating systems on mobile devices, Android has been the most frequently attacked, given that it is an open-source operating system (Teufl et al. 2014). Furthermore, although users are meant to download and install applications from the Android market, many obtain applications from unofficial, exterior markets, such as SlideME (2013). Quite surprisingly, the official Android market does not closely control its application content. A case in March 2011 found 50 Android market applications infected by DroidDream malware, which were

subsequently removed (Burguera et al. 2011). Likewise, in April 2013, security researchers identified malware that had been operating within Google Play for at least 10 months and infected 35 different applications. Google revealed that the infected applications had been downloaded between 2 and 9 million times (Arstechnica 2013). A report by F-Secure demonstrated that Android accounted for 79 % of malware in 2012, up from 66.7 % in 2011 and barely 11.25 % in 2010 (F-Secure 2013). As predicted by Trend Micro in March 2013, malicious Android applications will proliferate by 185 % in 2013. It is believed that adware and data theft are the most copious among such malicious applications (Hardwarezone 2013). In June 2013, Symantec discovered a malware that was holding a user's data at ransom. The latest ransomware demands $100 to unlock the device. The same author who released similar malware in June 2013 produced this most recent ransomware. Thus, the threat of mobile malware transcends merely stealing user data—it also emerges in the form of ransomware. It is worth noting that in this work, we focused on mobile bots, as 93 % of MalGenome samples are network-controlled bots (Yajin and Xuxian 2012). Therefore, bot and malware are used interchangeably.

Botnet is a network of infected devices under command of an attacker. The attacker, known as botmaster, is able to control the bots and command them to perform malicious activities such as stealing data, intercepting messages and making phone calls without users' knowledge. An infected computer and the hacker communicate through a rendezvous point that is called command and control (C&C) server. The first mobile bot was discovered in 2010 with the emergence of Geinimi attack (Lookout 2010).

With the aim of confronting malware, mobile devices have adopted traditional approaches such as the antivirus. This tactic is not as efficient (Sohr et al. 2011) against mobile device malware, as it requires continuous signature database updating and mobile malware is constantly modified to circumvent the various detection methods. For instance, the first version of DroidKungFu surfaced in June 2011, and it seemed one of the most sophisticated kinds of malware at that time. Shortly after, the second and third versions appeared in July and August, respectively. The fourth version was detected in October, and the fifth soon after that. Variants tend to utilize assorted encryption keys to protect themselves. Such malware adaptation indicates hackers' insistent attempts to bypass detection (Yajin and Xuxian 2012). Consequent to employing antivirus as a defense mechanism, mobile devices may end up with other threats including short message service (SMS) spam, personal data and user credential theft, content manipulation and unauthorized remote access. The SMS is in imminent danger as well, since people send text messages without any regard to security.

The intrusion detection system (IDS) is one of the solutions to diminishing threat impact, as it provides early detection and the prospect of warning users. Traditional approaches, such as the antivirus, are unable to offer early detection, and in most cases malware like bots populates mobile devices well prior to detection (García-Teodoro et al. 2009). In addition, an antivirus needs to monitor a device's activities closely and requires frequent signature update processes to detect new malware. These procedures demand excessive memory and power usage, which degrade mobile device performance (Lai and Liu 2011). Therefore, the aim in this paper is to evaluate the effectiveness of anomaly-based IDSs in malware detection using network traffic. There are two detection method types in IDSs: misuse-based and anomaly-based (Shamshirband et al. 2013). Unlike misuse-based, the anomaly-based IDSs do not necessitate signatures to detect intrusion. Besides, an anomaly-based IDS is capable of identifying unknown attacks based on the similar behaviour of other intrusions (Curiac and Volosencu 2012).

Machine learning (ML) classifiers have played a part in the development of intelligent systems for many years. MLs acquire a labelled dataset and produce a model as output, which can handle new data. Classifiers learn from an abundance of input and labelled output to build a model. As such, adopting machine learning classifiers is proven to enhance detection accuracy (Anuar et al. 2008). Subsequently, the intention in this work is to evaluate machine learning classifiers to provide an alternative solution to malware detection using network traffic. For an added advantage over built-in applications like antivirus solutions that drain a device's resources, in this study we capture network traffic from the exterior of mobile devices. In addition, the detection process can run using cloud services such as Patel et al. (2013), whereby network traffic is analyzed remotely. Such approach lowers resource consumption and software complexity (Oberheide et al. 2008).

Therefore, to evaluate IDS performance when detecting malware in mobile applications, we assess five classifiers in this study, namely the decision tree (J48), Bayes network (BN), multi-layer perceptron (MLP), k-nearest neighbours (KNN) and random forest. Using 1,000 malware samples from Android Malware Genome Project (Yajin and Xuxian 2012), this paper presents an extensive evaluation study with the purpose of demonstrating the effectiveness of malware detection using network traffic. This paper additionally assesses the performance of the detection process and contrasts it against other studies. To exhibit the model's effectiveness, the present work expands on evaluation studies by examining a private dataset, which consists of 30 new malware samples from 2013. A malware analyst from F-Secure Corporation supervised the process of malware analysis.

Since Android malware is a novel research area, there have been no tangible data samples available to the research community. Some researchers have produced their own malware and tested it on self-written malicious applications (Shabtai

et al. 2012). Others have attempted to collect thousands of applications using a script known as a crawler, which crawls the Internet and locates Android application files (Zheng et al. 2013). In 2012, researchers at the North Carolina State University published a comprehensive data sample called MalGenome that comprises of 49 different Android malware families with a total of 1,260 malware samples (Yajin and Xuxian 2012). A malware family is a collection of malware with similar behaviour. The aforementioned data samples were gathered between August 2010 and October 2011. We subsequently requested these data samples from the authors, since it is only available upon request from faculty members.

Quintessentially, investigating the dynamic behaviour of mobile applications is possible by monitoring a mobile device's system calls or network traffic. Establishing a network connection is an inevitable part of any mobile application, whether normal or malicious. Thus, examining the network traffic of a mobile device is a reasonable approach to detecting mobile malware. Furthermore, as useful as the host-based analysis is, many features need to be collected. As a result, volume and complexity of the dataset become an issue. In case of Android malware, system calls, API calls, native calls and user interaction such as screen touches and/or pushing a button needs to be collected. Some of those features require the device to be rooted. Whereas collecting network traffic is much easier and produces the same or better results. In addition, based on Felt et al. (2011), many malicious applications require network communication for their activity. Hence, the focus of this paper is also on probing the network traffic generated by mobile applications and utilizing machine learning classifiers to detect malware.

This paper presents a comprehensive study on evaluating the effectiveness of adopting machine learning classifiers to detect malware in mobile applications. Since the Android Operating System is the most popular system employed by end users, this research concentrates on Android applications. The contributions made by this work are as follows:

(a) The evaluation study applied 49 malware family types and 1,000 samples from the Android Malware Genome Project (Yajin and Xuxian 2012), which is considered an extensive malware data sample in mobile devices.
(b) Five machine learning classifiers were evaluated from several analysis aspects (i.e. logical-based, perceptron-based, static-based and instance-based techniques).
(c) The experiment employs 11 features in the evaluation study, which were extracted from network traffic generated by mobile applications. These features are categorized into four groups: basic information, content-based, time-based and connection-based features.
(d) An empirical validation conducted demonstrates the effectiveness of the selected classifiers.
(e) The practical evaluation carried out contains an additional private dataset consisting of the 30 latest malware from 14 malware family types to evaluate and validate the classifiers' performance. We gathered the most recent malware in 2013.

In this paper, the next section addresses related work on the topic. Section 3 thoroughly discusses the methodology and an overview of malware detection. The three main phases presented in Sect. 3 are data collection, features extraction and selection, and machine learning classifiers. Section 4 represents the evaluation studies as well as the simulation performance and experimental results. Finally, Sect. 5 comprises the concluding remarks and proposals for future research.

2 Related works

General solutions to detecting intrusions using IDSs involve using signature-based and anomaly-based detection (Verwoerd and Hunt 2002). The signature-based technique discovers intrusions using a predefined list of known attacks. Although this solution has the capacity to detect malware in mobile application, it requires constant updating of the predefined signature database. Additionally, it is less successful in detecting malicious activities using the signature-based method due to the rapidly changing nature of mobile malware (i.e. DroidKungFu). Based on Yajin and Xuxian (2012), current softwares can detect up to 79.6 % of mobile bots in our dataset, MalGenome. However, machine learning method, known as anomaly based, increases detection to over 90 %. An anomaly-based IDS method, on the other hand, adopts machine learning classifiers and is able to detect malware by learning their behaviour. It is feasible to model malware behaviour using a machine learning classifier and by employing the produced model to detect new malware. Machine learning (ML) classifiers have, for several years, served in developing intelligent systems by training machines on how to make decisions. With a dataset labelled as input, ML constructs a model that is applicable to new data to identify pattern similarities. Numerous studies with significant detection results have adopted a similar approach, with the intention of detecting intrusions effectively (Sangkatsanee et al. 2011; Zhao et al. 2012).

With the purpose of optimizing the process of classifying malware behaviour, it is necessary to collect specific malware features—something only achievable by investigating malware activities. There are two means of analyzing malware: static and dynamic analysis (Egele et al. 2008). Static analysis is the process of examining malware without executing mobile applications. For instance, a study was conducted whereby the requested and required permissions

were inspected to detect some Android malware (Huang et al. 2013). Similarly, a system was proposed to detect mobile threats to the homeland security via static analysis (Seo et al. 2013).

Static analysis is simple to implement, but produces less information, thus limiting the extraction of possible features from malware activities. In addition, attackers use various methods such as code obfuscation to evade detection through static analysis. As a result, to extract useful information and enrich the features, dynamic analysis serves to execute a mobile application and monitor its behaviour. A number of behaviour types that can be observed using dynamic analysis exist, such as the application used (Burguera et al. 2011), system calls (Dini et al. 2012), traffic generated and memory utilized.

Sarma et al. (2012) 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. Huang et al. (2013) used static analysis and evaluated four classifiers (i.e. AdaBoost, NB, DT48 and SVM) on application permissions to detect anomalies. They detected malware with a rate of 81 % using the Naïve Bayes classifier. Thus, they claimed that permission-based analysis is a quick filter for identifying anomalous applications. However, permission-based analysis is less effective regarding malware such as Basebridge, which hides an updated version within the original application and, as a result, slips into a mobile device without the user's knowledge and bypasses the permission. Thus, due to the aforementioned weaknesses in signature-based technique and static analysis, we opted to use anomaly-based technique and dynamic analysis in this paper.

Burguera et al. (2011) proposed Crowdroid, a dynamic analysis system that examines application behaviour to detect malware in Android by monitoring system calls. This framework is able to detect self-written malware (i.e. PJApps and Hong Toutou). Similarly, Shabtai et al. (2012) identified the best classification method out of six classifiers, namely DTJ48, NB, BN, k-Means, histogram and logistic regression, using the Andromaly framework. The framework adopts feature selection methods such as Chi-square, Fisher score and information gain to enhance the detection accuracy. As a result, they managed to achieve a 99.9 % accuracy rate with the decision tree (i.e. J48) classifier and information gain method. Unfortunately, although Shabtai et al. (2012) achieved great accuracy, they used self-written malware to test their framework on, which could have produced unrealistic results. In contrast, we used 1,000 real-world malware in this work instead of self-written malwares.

Dini et al. (2012) proposed a multi-level anomaly detector prototype by combining two system call levels: kernel and user level. With 12 system calls as the main features together with the K-nearest neighbours (KNN) classifier, Dini et al. successfully obtained a 93 % accuracy rate for 10 malwares. Although this approach is promising, it is incapable of detecting malware that avoids the system call with root permission, for example SMS malware that is invisible in the kernel (Security 2011). Therefore, network traffic analysis is used in this work to compensate the weakness of using system calls.

Zhao et al. (2012) proposed RobotDroid based on the SVM machine learning classifier to detect unknown malware in mobile devices. The focus was on privacy information leakage and hidden payment services. They evaluated three malware types, namely Gemini, DroidDream and Plankton. As a result, this framework is limited to few malware types and more would be required to increase detection accuracy.

Su et al. (2012) presented the smartphone dual defense protection framework, which has two phases: verification service and a network monitoring tool. The first, verification service phase, applies system call statistics to differentiate between malicious and normal codes in an application. The second phase monitors any possible malicious codes identified in the first phase. The two simulated classifiers to evaluate the proposed framework are J48 and random forest. The random-forest classifier achieved 96.67 % and the J48 classifier attained 91 % detection accuracy. However, this technique employed 32 malware families with only 180 samples, compared to 1,000 samples used in this research. Extra samples lead to results that are more accurate, because including as many samples as possible provides more comprehensive and reliable outcomes.

STREAM (Amos et al. 2013) was introduced in 2013 in which it collects a number of features such as battery, memory, network and permission. It then applied several machine learning classifier on the collected data. However, the authors used Android emulator to collect selected features, which was proved not as accurate as a real device (Raffetseder et al. 2007), and claimed that using a real device was impossible. In this work, we used a real device to conduct part of our experiment.

Hyo-Sik and Mi-Jung (2013) published their study in 2013 in which they applied machine learning on selected features of Android applications. The weakness of their work is that they evaluated their proposed system with five malware families, whereas we evaluated our system with 49 malware families, which in turn makes our result much reliable.

In 2014, DREBIN (Arp et al. 2014) was published in which it performs a broad analysis of Android applications to detect malwares. It collects permissions, hardware access, API calls, network address, etc. However, it is unable to detect malwares that use obfuscation technique and dynamically loaded code technique, whereas our work is fully capable of detecting such malwares.

The aforementioned approaches demonstrate the rationale behind applying machine learning classifiers to detect mobile malware. However, the prior evaluation is limited to a certain amount of malware, and few of the approaches consider feature selection in the classification process to increase result accuracy.

In Intrusion detection systems, accuracy rate plays a significant role in measuring the effectiveness of an approach. One of the motivations of this study is to increase the true-positive as well as reducing the false-positive rates beyond other studies. Based on the results section, the experiments run with machine learning classifiers successfully achieved superior detection rates.

3 Methodology

This section presents the overall workflow of the two experiments. Figure 1 illustrates the experiment workflow structure consisting of three phases. The first stage is data collection, which captures the network traffic of normal and malicious applications and transmits it to the next phase. In the second phase, which is feature selection and extraction, the TCP packets are filtered; then, features selected from among various network features are extracted, labelled and stored in a database to be applied in the next phase. The machine learning classifier entails the final phase, whereby the information in the database trains the machine learning classifier to produce a detection model. The following sections communicate each phase in detail.

3.1 Data collection phase

In this phase, as seen at the top of Fig. 1, different methods identify benign and malware network traffic. The methods consist of running an application on a device and capturing the network traffic, and using online dynamic analysis platforms. There are two reasons based on which we selected network traffic in this work. First, mobile bots have different intentions when it comes to host-based characteristics such as intercepting SMS, sending SMS, making phone calls, stealing passwords and hijacking an application. However, they all have a common feature, which is establishing network connection between the device and attacker. Second, we can extract abundant features from Android files such as permissions, java code, system calls and network traffic. However, the majority of applications, normal or malicious, require network connectivity. A recent work (Sanz et al. 2013) analyzed permissions of 2,000 applications. Over 93 % of malicious applications requested network connectivity. As a result, network access is requested from a majority of applications, particularly the malicious ones. Therefore, we opted to focus

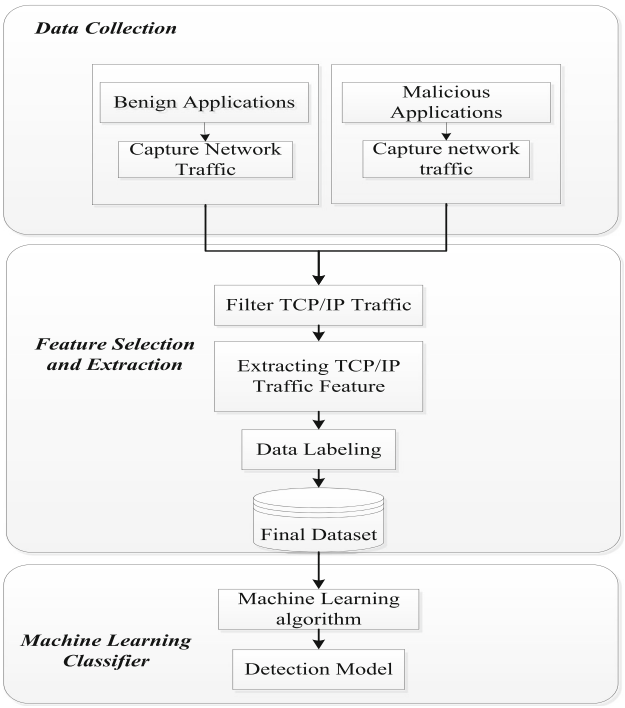

Fig. 1 The experiment work flow structure

on analyzing network traffic for effective Android malware detection. These techniques are described subsequently.

3.1.1 Benign applications

The network traffic of benign applications was captured on a real device using tPacketCapturePro (2013) on the Android operating system version 4.1.2 Jelly Bean (Android 2013). The top 20 free applications were selected from the Google Play market for benign traffic as shown in Table 1; play (2013). We captured the network traffic of the benign applications in an actual network environment and ran each application for 10, 20 and 30 min. To capture the network traffic, we ran the applications one by one with a filtering feature of PacketCapture Pro. The filter function enables filtering and removing the unwanted applications to ensure that tPacketCapture Pro only obtained the actual chosen applications. The reason behind running each application for particular amounts of time is to determine whether every application runs for the same length of time and to combine it with a malware dataset more easily. The 10-min running time was benign data with a 400-malware dataset and the combination of 20 and 30 min was for the 1,000-malware dataset. To guarantee that the selected applications were clean from any malware, the application developers' reputation was verified, and they appeared to be genuine application distributors.

Table 1 Normal applications selected (top free Google Play applications)

No.	Application	10 mins	20 mins	30 mins
1	We Chat	✓	✓	✓
2	Facebook	✓	✓	✓
3	LINE:Free Calls and Messages	✓	✓	✓
4	WhatsApp messenger	✓	✓	✓
5	CandyCrushsaga	✓	✓	✓
6	Pou	✓	✓	✓
7	Instagram	✓	✓	✓
8	Viber:FreeMesage and Call	✓	✓	✓
9	Facebook Messenger	✓	✓	✓
10	Go LauncherEx	✓	✓	✓
11	Cut the rope	✓	✓	✓
12	SubwaySurfers	✓	✓	✓
13	Skype-free IM and Videocalls	✓	✓	✓
14	Twitter	✓	✓	✓
15	Burger-BigFernand	✓	✓	✓
16	HardestGameEver2	✓	✓	✓
17	Temple Run 2	✓	✓	✓
18	Minion Rush	✓	✓	✓
19	Fruit Ninja Free	✓	✓	✓
20	Hill climb Racing	✓	✓	✓
	Total	200 mins	400 mins	600 mins

3.1.2 Malicious applications

This research incorporated public and private datasets to evaluate the selected classifier. MalGenome is a public dataset, which is a collection of 1,260 malwares categorized into 49 different families gathered between 2010 and 2011. It entails some of the most malicious mobile malwares in the world. Surprisingly, 14 out of 49 families in this set were collected from the official Android market known as Google Play. The private dataset is a collection of the latest mobile malware. Owing to the ongoing changes in mobile malware, which is the hackers' strategy to bypass the current detection methods, as well as the rapid increase of mobile malware, 30 malware samples were collected by our security team and supervised by a malware analyst. The bot samples in our experiment use network connectivity as their main communication means. However, some of them use SMS, not as the main communication means, but just to intercept incoming messages and leak their content to attackers via network traffic (Liang et al. 2013; Seo et al. 2013).

3.1.3 MalGenome dataset

MalGenome Project (2013) contains 1,260 malware data samples in 49 families, from which 1,000 samples from 49 malware families were selected to capture their network pattern. The reason for selecting 1,000 samples is that 93 % of all samples are bots. The samples were analysed with online dynamic analysis platforms, namely Anubis iseclab Anubis (2013) and SandDroid (2013). Dynamic online analysis platforms run an application in a controlled environment and capture the network traffic, which will be used for research purposes. Among 1,260 samples in the MalGenome dataset, Anubis iseclab generated 1,007 samples of traffic and SandDroid generated 164. These dynamic analysis platforms capture network traffic in a monitored environment. Since malware samples do not require installation on a physical device to generate traffic, this approach reduces the time required for traffic generation. Out of 1,260 files available in MalGenome, 93 % are bots, which are 1,171 files. We checked for valid server throughout our experiment and removed 171 files with unresponsive server. We made sure that the three-way handshake is available in traffic, which is a sign of a working server.

3.1.3.1 Latest mobile malware dataset For further investigation, we conducted another experiment with 14 mobile malware families that contain 30 malware samples. We collected the latest malware data samples from 2013. The network traffic generated by each sample was acquired using the same method as for the MalGenome data sample.

3.2 Feature selection and extraction phase

Wireshark and Java routine filtered benign and malware network traffic gathered during the data collection phase to remove unwanted packets from among the captured packets. *Domain name system* (DNS) packets were filtered out from the network traffic data; then only TCP packets were used in the dataset since the conversation between a mobile malware and a hacker is conveyed through TCP packets. Thereafter, all related information was extracted using tshark (2013), which is a terminal-based version of *wireshark* that was used to extract 11 features from the TCP packets (Table 2). The features were chosen from numerous network features in the packet-level features of the TUIDS intrusion dataset (Gogoi 2013). Above all, the main challenge in feature selection was finding the most relevant features that led to the highest true positive rate. A large number of features in the dataset should be filtered and refined. In addition, some features correlate to each other, and this hinders the intrusion detection process. Moreover, some features may contain redundant information from other features. Redundant features increase computational time and reduce IDS accuracy. A specific method called *ClassifierSubsetEval* from Weka machine learning tool was applied to attribute selection. We selected six features out of 11 after applying feature selection algorithm based on the results. It is worth noting that there is

Table 2 List of extracted network traffic features for our dataset

Label/feature name		Type	Description
Basic features			
1	src-ip	c	Source IP address
2	dst-ip	c	Destination IP address
3	src-port	c	Source host port number
4	dst-port	c	Destination host port number
5	fr-len	c	Frame length
6	fr-no	c	Frame number
Content-based features			
7	num-get/post	c	http protocol used to submit data from client to server
Time-based Features			
8	count-serv-src	c	Number of frames received by unique destination in the last T seconds from the same source
9	count-serv-dst	c	Number of frames received by unique source in the last T seconds from the same destination
Connection-based features			
10	num-packets-src-dst	c	The number of packets flowing from source to destination
11	num-packets-dst-src	c	The number of packets flowing from destination to source

c Continuous

no specific feature that can make a difference between benign apps and malicious apps, but a group of features. A combination of all the selected features helped to find anomalies, not just a single feature. As an instance, specific IP addresses in China are related to malicious activities. In addition, because of leaking data, the frame length is larger than normal frames. As a result the combination of both features results in anomaly detection. Classifiers study the applications based on a group of features and build a behaviour pattern to detect the malware. Any behaviour pattern that deviates from a clean and normal pattern will be regarded as malignant.

More specifically, the features selected for our dataset are derived from four feature categories (Lee and Stolfo 2000). The categories are *basic features*, *content based*, *time based* and *connection based*. The basic features class is the most prevalent network traffic feature employed by numerous researchers for malware detection. This category contains the fundamental information of network traffic, such as source IP address, destination IP address, source host port number, destination host port number frame length and frame number. The second category of content-based features is a frequent pattern of connection records. The time-based feature in the third category keeps record of a connection in the past 10 s with the same destination host as that of the current connection. The fourth group, or the connection-based features, counts the number of continuous data packets from source to destination and vice versa. In the present study, 11 selected features, as shown in Table 3, were fed into our dataset for malware detection.

The extracted features were stored as a sequence of *comma separated values* (CSV) files. Each record consists of the summary data of 11 network features. Experts in intrusion detection labelled the dataset information as either '*normal*' or '*infected.*' The labelling process made use of a malware dataset considered '*infected*' and a benign dataset categorized '*normal.*' Finally, the ultimate dataset meant the integration of malware with normal datasets. Then, one of the features in the Weka pre-processor *weka.filters.unsupervised.instance.randomize* shuffled the records in the final dataset. Such data randomization ensures experiment validity.

3.3 Machine learning classifier phase

In the final stage, that is the machine learning classifier phase, the classifiers' output is produced. This phase determines the finest machine learning classifier for malware detection based on results. Two sub-datasets from MalGenome were used to run experiments with five selected classifiers. The first and second sub-datasets consist of 49 malware families with 400 samples and 1,000 samples, respectively.

In spite of several machine learning techniques, a review by Kotsiantis et al. (2006) categorized machine learning based on artificial intelligence (logic-based techniques, perceptron-based techniques) and statistics (Bayesian network, instance-based techniques) as demonstrated in Table 4. This section presents the concepts and descriptions of our selected classifier applied in the current experiment.

- *Random forest*: Random forest is a blend of random subspace and bagging method proposed by Tin Kam (1998). Random forest ensembles classifiers using many decision tree models. A different subset of training data is selected with a replacement to train each tree. The remaining training data serves to estimate the error and variable importance (Breiman 2001). This classifier is a logic-based algorithm that is proved to produce high-accuracy result as in Eskandari and Hashemi (2012); Su et al. (2012) for malware detection.
- *J*48: J48 is a predictive machine learning classifier that decides the target value of a new sample based on various attribute values of the available data. This classifier consists of dependent and independent variables. The dependent variable is the attribute to be predicted, while the independent variable is the attribute that helps predict the value of the dependent variable. To classify new samples, a decision tree based on the attribute values in

Table 3 The dataset sample

Frame length	Frame number	Srcport	Dstport	Decimal IP src	Decimal IP Dst	Get/post number'	Count-serv-src	Count-serv-dst	Num-pack-src-dst	Num-packs-dst-src	Label
378	6351	55391	80	167772687	3634170898	12	5	3	81	80	Normal
54	65532	80	48354	1249740646	167772687	0	5	5	665	623	Normal
1,254	227780	52593	5555	167772674	167772687	0	7	7	82828	81765	Infected
112	250490	5555	53344	167772687	167772674	0	3	4	71024	71123	Infected
54	46769	43123	80	167772687	921072993	0	4	2	1613	1625	Normal
1,178	47541	80	39383	2915190176	167772687	0	6	6	118	105	Infected
54	133525	35430	5555	167772674	167772687	0	6	6	28357	28398	Infected
54	48639	41142	80	167772687	1158721322	0	5	1	107	115	Normal
54	31976	43759	80	167772687	921072708	0	5	3	14	17	Normal
404	20266	42718	80	167772687	3689567708	10	5	1	1287	1346	Infected

the training set must be created. These attribute values determine data instances to classify and have the most information. Shabtai et al. (2012) applied this classifier and obtained very high detection results. J48 is also a type of logic-based learning.

- *MLP:* Multi-layer perceptron is a feed forward artificial neural network model. MLP consists of multiple layers of nodes that interact via weighted connections (Pal and Mitra 1992).
- B*ayesian Net*: Bayesian, or the Bayes network, is a probabilistic graphical model that represents a set of random variables. A Bayesian network models the flow of information in decision tables while reasoning on new cases. Necessary probabilities can calculate directly from training data by substituting the foregoing decision value in the loop (Friedman et al. 1997). This classifier is the most well-known representation of a statistical learning algorithm.
- *KNN*: K-nearest neighbours is one of the simplest classification techniques, with less or no prior knowledge of data distribution. Typically, the classification involves partitioning samples into training and testing processes. During training, each training sample only uses true class to train the classifier, while testing aims to predict the class. The predicted test sample class is set equal to the true class among the nearest training samples (Liao and Vemuri 2002). In addition, KNN is a type of instance-based learning known as lazy learning classification. In this experiment, three neighbours comprised the KNN setting to perform the classifier.

The experiments were performed on a desktop computer with Intel core i73770 CPU of 3.40 GHZ and 4 GB of RAM. The operating system is Microsoft Windows 7 Professional. The experiments employed the *Waikato Environment for Knowledge Analysis* (WEKA) machine learning tool due to its simplicity and user-friendly environment. The selected classifiers functioned in a default setting, except KNN with three neighbours. The Java Runtime Environment (JRE) version 1.7.0_21 operated the WEKA software.

4 Experiments and results

This section presents the experimental results and performance evaluation of malware detection. The assessment entailed two experiments, the first of which utilized the public dataset MalGenome and the second was done on a private dataset. The MalGenome experiment used k-fold cross validation, otherwise known as the tenfold method. K-fold cross validation makes good use of the holdout scheme and runs, repeatedly, k-fold times. The dataset has two segments, which are k subsets as the test set and k-1 subsets as the training

Table 4 Summary of pros and cons of the classifier categories

Class	Sub-class	Classifier	Pro	Con
Artificial intelligence	Logic-based	Random forest	Often improves predictive performance	Usually produce output that is hard to analyze
		DTJ48	Fast and scalable classifier of decision tree comprehensibility	Less effective on predicting the value of continuous class attribute
	Perceptron-based	MLP	Parelellization of learning process and accurate estimation	Large computational time for classification
Statistics	Bayesian	BN	Fast and efficient computation, and quick data training	Impractical for dataset with many features
	Instance-based	KNN	More robust than MLP when using small k value	Large storage require,large computational time for classification

set. Finally, the average of all k trials is computed to make an evaluation (Schneider 1997). Meanwhile, the latest malware experiment used the training set and test set option in WEKA. A training set is a set of data employed in various areas to discover the potential for a predictive relationship. The test set plays an important role in examining how effective the classifier is. It is worth noting that the test dataset is not included in the training dataset. In the end, following the experimental results for both situations, the ideal classifier was determined.

4.1 Evaluation measures

To evaluate the detection performance successfully, it is necessary to identify appropriate performance metrics. The following measures were derived from the confusion matrix to calculate and be applied to classifier evaluation.

$$\text{True positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

$$\text{False positive rate} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{5}$$

where TPR, or the true-positive rate, is the value of predicted malware classified correctly. FPR is the false-positive rate value of normal data incorrectly predicted as malware. TP, or true positive, is the number of normal data correctly classified. TN, true negative, is the number of malware samples correctly classified. FP, false positive, is the number of normal samples classified as malware. FN, false negative, is the number of malware samples classified as normal. Precision, which is also called positive-predictive value, returns the rate of relevant results rather than irrelevant results. Recall is the sensitivity for the most relevant result. F-Measure is the value that estimates the entire system performance by combining precision and recall into a single number. The maximum value of 1.000 indicates the best result.

4.2 MalGenome experiments and results

The results obtained from a tenfold validation test using five selected classifiers to perform the experiments are presented in Table 5. The table shows the performance of each classifier in two experiment sets for malware detection. Classifier performance is measured with five evaluation metrics, namely *TPR*, *FPR*, precision, recall and f-measure.

Experiment I, conducted using 400 samples, demonstrates that the multi-layer perceptron classifier produced a higher FPR result with 5.17 % compared to the random forest with 0.04 %. This indicates that the MLP classifier is less effective than other selected classifiers for malware detection. Random forest also attained an elevated value for precision, recall and f-measure of 1.000.

In Experiment II, carried out using 1,000 samples, *BN* and random forest obtained 99.97 %, which is the highest *TPR* value in malware detection. Meanwhile, dataset size also affects malware detection effectiveness. The values for *BN, MLP* and *KNN TPR* decrease from that in experiment I, but the *J*48 classifier does not experience any result changes. *BN* and random forest gained the maximum value for precision, recall and f-measure, which increased as the dataset size increased. High recall indicates that an algorithm returned the most relevant results, while high precision means that an algorithm provided substantially more relevant results than irrelevant results.

Table 5 Malware detection evaluation result

Experiment	Classifier	TPR (%)	FPR (%)	Precision	Recall	F-measure
I	BN	99.88	0.12	0.999	0.999	0.999
	MLP	94.83	5.17	0.949	0.948	0.948
	J48	99.90	0.10	0.999	0.999	0.999
	KNN	98.73	1.27	0.987	0.987	0.987
	RandForest	99.96	0.04	1.000	1.000	1.000
II	BN	99.97	0.03	1.000	1.000	1.000
	MLP	93.03	6.97	0.932	0.972	0.944
	J48	99.90	0.10	0.999	0.999	0.999
	KNN	98.63	1.37	0.986	0.986	0.986
	RandForest	99.97	0.34	1.000	1.000	1.000

Owing to resource restraints on mobile devices, smartphone gadgets require running lightweight processes. However, such lightweight processes may affect the accuracy and effectiveness of Android malware detection. The current solutions for malware detection on smartphones literally consume high resources (memory, battery, CPU and storage) during operation. Thus, the time spent by classifiers plays a crucial role, as it affects resource consumption on a mobile device. The experiments performed used varying numbers of selected features. Choosing suitable features entailed two approaches: using the feature selection method in the WEKA software, and selecting features manually without the intervention of WEKA. Upon feature selection implementation in WEKA software, the number of features was reduced from 11 to 6.

Table 6 displays a processing time comparison between using feature selection in WEKA and without feature selection during the experiments. Evidently, the time to build the model improved with a decreased number of features. When all 11 features took part in the experiment, the simulator produced the model in 7.3 s, whereas with feature selection, the time reduced to 4.07 s for the Bayes network classifier. However, the other classifiers also displayed a decrease in processing time following feature selection application. Thus, processing time to build a model depends on dataset size, whereby with larger dataset sizes, processing time increases.

Table 7 lists the experimental results of applying classifiers with and without feature selection to our dataset. The TPR results obtained increased for four classifiers out of five when using the feature selection method. Random forest demonstrated the best results compared to all other classifiers.

As formula 3 in Sect. 4.1 displays, precision is the percentage of instances identified correctly from all data. In other words, precision is how strong a classifier is in terms of predicting positive instances. The random forest classifier achieved a TPR of 99.99 % and precision of 1.000, meaning that it can predict positive instances very well.

Table 6 Comparison of processing time (in seconds)

Classifier\sample	I	II
Processing time without feature selection (s)		
Bayes network	7.3	22.94
Multi-layer perceptron	303	761.28
J48	18.2	100.53
K-nearest neighbours	0.09	1.08
Random forest	20.67	118.05
Processing time with feature selection		
Bayes network	4.07	10.66
Multi-layer perceptron	158.88	375.15
J48	6.62	52.24
K-Nearest neighbours	0.05	0.18
Random forest	15.76	62.37

4.3 Latest malware experiment and results

The previous experiment addressed tenfold cross validation. However, the new malware dataset falls into two datasets, training and testing. Meanwhile, the testing and training set results were compared to obtain performance accuracy. Table 8 displays the experiment's TPR, FPR, precision, recall and f-measure results. Surprisingly, the highest accuracy achieved was 84.57 % with the KNN classifier, followed by MLP with 83.97 %, BN with 75.63 %, random forest with 74.15 % and J48 with 73.85 %. KNN also produced the greatest precision, recall and f-measure value. Although the random forest had the best TPR result in the prior experiment for the MalGenome dataset, it only achieved 74.15 % on the latest malware dataset.

4.4 Result comparison

To substantiate the result effectiveness, the two comparisons made are as follows:

Su et al. (2012) conducted a study on mobile malware, in which they defined two levels of detection. The first level examines the mobile operating system's calls and determines an application's maliciousness through a statistical approach. The malicious applications identified in level one are re-examined in level two, where probing the applications' network traffic takes place. The following nine statistical features are extracted: the average and standard deviation of the number of sent/received packets, the average and standard deviation of the number of bytes sent/received and the average TCP/IP session duration. Su et al. employed the same, MalGenome data sample, as in this research. The two machine learning classifiers used are J48 and random forest. Su et al. (2012) chose only 55 malicious samples compared to

Table 7 Comparison with and without feature selection

Experiment	Classifier	Without feature selection				
		TPR (%)	FPR (%)	Precision	Recall	F-measure
I	BN	99.88	0.12	0.999	0.999	0.999
	MLP	94.83	5.17	0.949	0.948	0.948
	J48	99.90	0.10	0.999	0.999	0.999
	KNN	98.73	1.27	0.987	0.987	0.987
	Random forest	99.96	0.04	1.000	1.000	1.000
II	BN	99.97	0.03	1.000	1.000	1.000
	MLP	93.03	6.97	0.932	0.972	0.944
	J48	99.90	0.10	0.999	0.999	0.999
	KNN	98.63	1.37	0.986	0.986	0.986
	Random forest	99.97	0.34	1.000	1.000	1.000
Experiment	Classifier	With feature selection				
		TPR (%)	FPR (%)	Precision	Recall	F-measure
I	BN	99.97	0.03	1.000	1.000	1.000
	MLP	91.73	8.27	0.921	0.917	0.918
	J48	99.98	0.02	1.000	1.000	1.000
	KNN	99.65	0.35	0.996	0.996	0.996
	Random forest	99.98	0.02	1.000	1.000	1.000
II	BN	99.98	0.02	1.000	1.000	1.000
	MLP	88.25	11.75	0.887	0.883	0.88
	J48	99.98	0.02	1.000	1.000	1.000
	KNN	99.65	0.35	0.997	0.997	0.997
	Random forest	99.99	0.01	1.000	1.000	1.000

Table 8 Latest malware detection evaluation results

Classifier	TPR (%)	FPR (%)	Precision	Recall	F-measure
Latest Malware result					
Bayes network	75.63	24.37	0.841	0.756	0.75
Multi-layer perceptron	83.97	16.03	0.88	0.84	0.839
J48	73.85	26.15	0.836	0.738	0.73
K-nearest neighbours	84.57	15.43	0.884	0.846	0.845
Random forest	74.15	25.85	0.837	0.741	0.734

1,000 samples employed in the current paper. The J48 classifier achieved 91.6 % and the random forest achieved 96.7 % detection rate accuracy, whereas the highest detection rate attained in this research was 99.99 % with the random forest classifier. The results signify the comprehensiveness and efficacy of this study.

Yerima et al. (2013) carried out a research work on mobile malware with the Bayesian classifier. They utilized a MalGenome data sample for the dataset and conducted several experiments on the sub-datasets of MalGenome. Their experiments had 100, 250, 500, 1,000 and 1,600 samples. Table 9 lists the results attained in their experiments.

Table 9 Results from a similar study

No. of samples	True positive rate (TPR) (%)
100	84.40
250	88.60
500	89.00
1,000	89.20
1,600	90.60

The highest detection rate from Yerima et al.'s work was 90.60 % compared to 99.99 % in the present study. If the number of samples was considered, the result would be

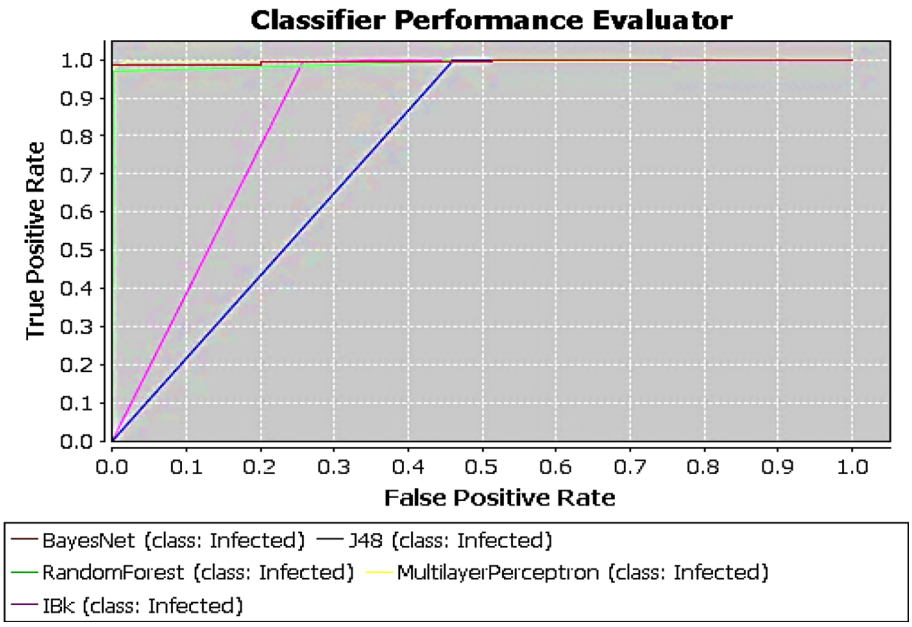

Fig. 2 Comparison of ROC curve for five classifiers on the latest malware

89.20 % for 1,000 samples against the best result of this study (99.99 %). Overall, the present research work transcends the mentioned related works in terms of comprehensiveness and results.

4.5 ROC curve

The receiver operating characteristic curve (Metz 1978), also known as the ROC curve, is a plot of performance measure that is determined by the true-positive rate versus the false. ROC is mainly used for visualizing, organizing and selecting the optimal classifier based on the varying classification algorithm performances (Fawcett 2006). Spackman (1989) adopted ROC graphs in machine learning and is one of the first to demonstrate the significance of ROC curves in evaluating and comparing algorithms. In recent years, the increasing application of ROC curves in the machine learning research community to measure performance is quite obvious (Bradley 1997). ROC analysis has found ramifications into sequence matching comparisons (Gribskov and Robinson 1996), comparing the best average configuration in experiments (Shabtai et al. 2012), visualizing and analyzing the behaviour of diagnostic systems (Lamiaa Ibrahim et al. 2013) and evaluating classifier accuracy (Kolter and Maloof 2006).

The area below the ROC curve, known as AUC, is widely utilized for weighing classifier performance on the latest malware dataset with the following defined levels:

- 1.0: perfect prediction
- 0.9: excellent prediction
- 0.8: good prediction
- 0.7: mediocre prediction
- 0.6: poor prediction
- 0.5: random prediction
- < 0.5: poor prediction.

Figure 2 illustrates our experimental ROC curve plot of the selected classifiers for the latest malware dataset described in section 3.1.2.2 with 11 features. The result obtained in 4.3 serves as input to form ROC curves. In each of the ROC plots, the X-axis represents the true-positive rate (TPR) calculated as the percentage of data correctly detected as malware. A data point in the upper left corner corresponds to optimal, high performance, i.e. high detection rate with low false alarm. The closer the curve is to the left and top borders of the ROC space, the more accurate the detection rate is. Clearly, J48 outperformed all other classifiers. This means that the J48 very rarely predicts any cases as positive, and this "overcaution" leads to what appears to be acceptable accuracy. Other classifiers, such as MLP, random forest, and BN plotted well provide the best malware detection average. Nevertheless, the ROC curve for KNN dominates over all other classifiers with respect to the latest malware detection as well as in terms of matching the predicted experimental performance involving a larger collection of executable. KNN is dependent on the number of neighbors, K, in the algorithm.

Table 10 represents the areas under the curve for the classifiers used in this experiment.

As the table illustrates, the Bayes network and multi-layer perceptron classifiers provide the best AUC value with 0.995, which signifies excellent prediction. The random forest classifier comes next with 0.991, denoting excellent prediction as well. The k-nearest neighbour value achieved was 0.869, which is good prediction. Finally, the J48 classifier attained 0.77, or mediocre prediction. Overall, the ROC curve and

Table 10 AUC values for the latest malware

Classifier	AUC
Bayes network	0.995
Multi-layer perceptron	0.995
J48	0.77
K nearest neighbours	0.869
Random forest	0.991

AUC values confirm that the most recent malware experiment provides compelling results regarding malicious application detection.

5 Conclusions and future work

This study presented an evaluation using machine learning classifiers to detect mobile malware effectively by selecting appropriate network features for inspection by the classifiers, as well as to determine the ideal classifier based on true-positive rate (TPR) values. The significance of this research lies in having applied the newest, together with self-collected, samples. Additionally, the classifiers' results and performance proved quite compelling.

In this research, we evaluated various machine learning classifiers to enhance the malware detection outcome for a large collection of file samples and obtain the optimum classifier able to detect mobile malware. The classifiers selected were Bayes network, multi-layer perceptron, decision tree (J48), K-nearest neighbour and random forest.

The MalGenome Project samples in our experiment comprised 49 different families containing 1,260 Android malware samples, but only 1,000 were used. The machine learning process had three phases: (1) data collection, which captured network traffic; (2) feature selection and extraction; and (3) the machine learning classifier. For the normal dataset, we selected the top 20 free applications from Google Play.

The experimental results indicate 99.99 % detection rate accuracy with the random forest classifier for the Genome Malware dataset. Hence, higher correct classification percentages from the recent malware detection method demonstrate the effectiveness of the KNN machine learning technique with 84.57 %. It also proves that machine learning classifiers are able to detect the latest malware. The larger the size of the dataset, the higher is the processing time to build the detection model and increase the precision, recall and f-measure value.

The current work is a baseline for future research in which researchers utilize game theory and multi-agent systems for detection. Therefore, the system is dynamic and attackers are not able to bypass the system, as it has been done before in Shamshirband et al. (2014a, b).

As future work, developing real-time mobile malware detection using machine learning classifiers in the cloud is suggested. Nevertheless, this approach proved the effectiveness and efficiency of machine learning in real mobile malware operational environment.

Additionally, it is useful to conduct a research on why the selected features yielded such high results. Furthermore, the relation between features can be examined to know their influence and role in the detection system. In a similar work (Shabtai et al. 2014), the authors tried to find the best subset of features from a pool of features by using experiments. Such research works result in more robust detection systems, as there is still a need to further examine the features.

Acknowledgments The authors would like to thank Hamid Talebian and Shahaboddin Shamshirband for their valuable comments. This work is supported by the Ministry of Science, Technology and Innovation, Malaysia under Grant eScienceFund 01-01-03- SF0914.

References

Amos B, Turner H, White J (2013) Applying machine learning classifiers to dynamic android malware detection at scale. In: Proceedings of the 9th international wireless communications and mobile computing conference (IWCMC), Sardinia, Italy, pp 1666–1671

Android (2013) Android 4.2, Jelly Bean. http://www.android.com/about/jelly-bean/. Accessed June 2013

Anuar NB, Sallehudin H, Gani A, Zakaria O (2008) Identifying false alarm for network intrusion detection system using hybrid data mining and decision tree. Malays J Comput Sci 21(2):101–115

Anubis (2013) Anubis: analyzing unknown binaries. http://anubis.iseclab.org/. Accessed Feb 2013

Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K (2014) DREBIN: effective and explainable detection of android malware in your pocket. In: Proceedings of the 2014 network and distributed system security (NDSS) symposium, San Diego, USA (2014)

Arstechnica (2013) More BadNews for android: new malicious apps found in google play. http://arstechnica.com/security/2013/04/more-badnews-for-android-new-malicious-apps-found-in-google-play/. Accessed 1st Jan 2013

Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Burguera I, Zurutuza U, Nadjm-Tehrani S (2011) Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices, Chicago, pp 15–26

Burguera I, Zurutuza U, Nadjm-Tehrani S (2011) Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices, Chicago, USA, pp 15–26

Curiac D-I, Volosencu C (2012) Ensemble based sensing anomaly detection in wireless sensor networks. Exp Syst Appl 39(10):9087–9096

Dini G, Martinelli F, Saracino A, Sgandurra D (2012) MADAM: a multi-level anomaly detector for android malware. In: Proceedings of the 6th international conference on mathematical methods, models and architectures for computer network security (MMM-ACNS 2012), Saint Petersburg, Russia, pp 240–253

Egele M, Scholte T, Kirda E, Kruegel C (2008) A survey on automated dynamic malware-analysis techniques and tools. ACM Comput Surv 44(2):1–42

Eskandari M, Hashemi S (2012) A graph mining approach for detecting unknown malwares. J Vis Lang Comput 23(3):154–162

Fawcett T (2006) An introduction to ROC analysis. Pattern Recognit Lett 27(8):861–874

Felt AP, Finifter M, Chin E, Hanna S, Wagner D (2011) A survey of mobile malware in the wild. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices, Chicago, Illinois, USA, pp 3–14

Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Mach Learn 29(2–3):131–163

F-Secure (2013) Android accounted for 79% of all mobile malware in 2012, 96% in Q4 alone. http://techcrunch.com/2013/03/07/f-secure-android-accounted-for-79-of-all-mobile-malware-in-2012-96-in-q4-alone/. Accessed 1st June 2013

García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur 28(1–2):18–28

Gogoi P, Bhattacharyya DK, Borah B, Kalita JK (2013) MLH-IDS: a multi-level hybrid intrusion detection method. Comput J 2013 doi:10.1093/comjnl/bxt044. Online. http://comjnl.oxfordjournals.org/content/early/2013/05/12/comjnl.bxt044.abstract. Accessed 12 May 2013

Gribskov M, Robinson NL (1996) Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. Comput Chem 20(1):25–33

Hardwarezone (2013) Trend micro predicts android malware increase by 185%. http://www.hardwarezone.com.ph/tech-news-trend-micro-predicts-android-malware-increase-185. Accessed 1st Jan 2013

Huang C-Y, Tsai Y-T, Hsu C-H (2013) Performance evaluation on permission-based detection for android malware. In: Pan, J-S, Yang C-N, Lin C-C (eds) Advances in intelligent systems and applications, vol 2. Springer, Berlin, pp 111–120

Hyo-Sik H, Mi-Jung C (2013) Analysis of android malware detection performance using machine learning classifiers. In: Proceedings of the international conference on ICT convergence (ICTC), Jeju, Ethiopia, pp 490–495

Kolter JZ, Maloof MA (2006) Learning to detect and classify malicious executables in the wild. J Mach Learn Res 7:2721–2744

Kotsiantis SB, Zaharakis ID, Pintelas PE (2006) Machine learning: a review of classification and combining techniques. Artif Intell Rev 26(3):159–190

Lai Y, Liu Z (2011) Unknown malicious code detection based on bayesian. Procedia Eng 15:3836–3842

Lamiaa Ibrahim MS, Rahman Azema Abd El, Zeidan Amany, Ragb Maha (2013) Crucial role of CD4+CD 25+ FOXP3+ T regulatory cell, interferon-γ and interleukin-16 in malignant and tuberculous pleural effusions. Immunol Investig 42(2):122–136

Lee W, Stolfo SJ (2000) A framework for constructing features and models for intrusion detection systems. ACM Trans Inf Syst Secur 3(4):227–261

Liang S, Keep AW, Might M, Lyde S, Gilray T, Aldous P, Horn DV (2013) Sound and precise malware analysis for android via pushdown reachability and entry-point saturation. In: Proceedings of the third ACM workshop on security and privacy in smartphones & mobile devices, Berlin, Germany, pp 21–32

Liao Y, Vemuri VR (2002) Use of k-nearest neighbor classifier for intrusion detection. Comput Secur 21(5):439–448

Lookout (2010) Security alert: geinimi, sophisticated new android trojan found in wild. https://blog.lookout.com/blog/2010/12/29/geinimi_trojan/. Accessed 1st July 2014

Metz CE (1978) Basic principles of ROC analysis. Semin Nucl Med 8(4):283–298

Oberheide J, Veeraraghavan K, Cooke E, Flinn J, Jahanian F (2008) Virtualized in-cloud security services for mobile devices. In: Proceedings of the 1st workshop on virtualization in mobile computing, Breckenridge, Colorado, pp 31–35

Pal SK, Mitra S (1992) Multilayer perceptron, fuzzy sets, and classification. IEEE Trans Neural Netw 3(5):683–697

Patel A, Taghavi M, Bakhtiyari K (2013) An intrusion detection and prevention system in cloud computing: a systematic review. J Netw Comput Appl 36(1):25–41

Play G (2013) Shop android apps. https://play.google.com/store?hl=en. Accessed February 2013

Project MG (2013) Android malware genome project. http://www.malgenomeproject.org/. Accessed Feb 2013

Raffetseder T, Kruegel C, Kirda E (2007) Detecting system emulators. In: Proceedings of the 10th international conference ISC, Valparaíso, Chile, pp 1–18

SandDroid (2013) SandDroid-an APK analysis sandbox. http://sanddroid.xjtu.edu.cn/. Accessed April 2013

Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C (2011) Practical real-time intrusion detection using machine learning approaches. Comput Commun 34(18):2227–2235

Sanz B, Santos I, Laorden C, Ugarte-Pedrero X, Nieves J, Bringas PG (2013) MAMA: manifest analysis for malware detection in android. Cybern Syst 44(6–7):469–488

Sarma BP, Li N, Gates C, Potharaju R, Nita-Rotaru C and Molloy I (2012), "Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM symposium on access control models and technologies, Newark, New Jersey, USA, pp 13–22

Schneider J (1997) Cross validation. http://www.cs.cmu.edu/~schneide/tut5/node42.html. Accessed July 2013

Security P (2011) Rootkits: almost invisible malware. http://www.pandasecurity.com/homeusers/security-info/types-malware/rootkit/. Accessed July 2013

Seo S-H, Gupta A, Mohamed Sallam A, Bertino E, Yim K (2013) Detecting mobile malware threats to homeland security through static analysis. J Netw Comput Appl doi:10.1016/j.jnca.2013.05.008. Online. http://www.sciencedirect.com/science/article. Accessed Oct 2013

Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y (2012) Andromaly: a behavioral malware detection framework for android devices. J Intell Inf Syst 38(1):161–190

Shabtai A, Tenenboim-Chekina L, Mimran D, Rokach L, Shapira B, Elovici Y (2014) Mobile malware detection through analysis of deviations in application network behavior. Comput Secur 43:1–18

Shamshirband S, Anuar NB, Kiah MLM, Patel A (2013) An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique. Eng Appl Artif Intell 26(9):2105–2127

Shamshirband S, Anuar NB, Kiah MLM, Rohani VA, Petković D, Misra S, Khan AN (2014) Co-FAIS: cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks. J Netw Comput Appl 42:102–117

Shamshirband S, Patel A, Anuar NB, Kiah MLM, Abraham A (2014) Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks. Eng Appl Artif Intell 32:228–241

SlideME (2013) SlideME | android apps market: download free & paid android application. http://slideme.org/. Accessed 1st Oct 2013

Sohr K, Mustafa T, Nowak A (2011) Software security aspects of Java-based mobile phones. In: Proceedings of the 2011 ACM symposium on applied computing, Taichung, Taiwan, pp 1494–1501

Spackman KA (1989) Signal detection theory: valuable tools for evaluating inductive learning. In: Proceedings of the 6th international

workshop on machine learning, Ithaca, New York, USA, pp 160–163

Su X, Chuah M, Tan G (2012) Smartphone dual defense protection framework: detecting malicious applications in android markets. In: Proceedings of the mobile ad-hoc and sensor networks (MSN), 2012 eighth international conference on, Chengdu, China, pp 153–160

Survey G (2013) Our mobile planet: global smartphone user. http://services.google.com/fh/files/blogs/final_global_smartphone_user_study_2012.pdf. Accessed June 2013

Symantec (2013) Android ransomware predictions hold true. http://www.symantec.com/connect/blogs/android-ransomware-predictions-hold-true. Accessed 1st Sept 2013

Teufl P, Ferk M, Fitzek A, Hein D, Kraxberger S, Orthacker C (2013) Malware detection by applying knowledge discovery processes to application metadata on the Android Market (Google Play). In: Security and communication networks. doi:10.1002/sec.675 [Online]. http://dx.doi.org/10.1002/sec.675. Accessed 1st April 2014

Tin Kam H (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844

tPacketCapturePro (2013) tPacketCapture-Capture Communication Packets. http://www.taosoftware.co.jp/en/android/packetcapture/. Accessed April 2013

tshark (2013) tshark-the wireshark network analyzer. http://www.wireshark.org/docs/man-pages/tshark.html. Accessed Feb 2013

Verwoerd T, Hunt R (2002) Intrusion detection techniques and approaches. Comput Commun 25(15):1356–1365

Yajin Z, Xuxian J (2012) Dissecting android malware: characterization and evolution. In: Proceedings of the 2012 IEEE symposium on security and privacy (SP), San Fransico, USA, pp 95–109

Yerima SY, Sezer S, McWilliams G, Muttik I (2013) A new android malware detection approach using bayesian classification. In: Proceedings of the 2013 IEEE 27th international conference on advanced information networking and applications (AINA), Barcelona, Spain, pp 121–128

Zhao M, Zhang T, Ge F, Yuan Z (2012) RobotDroid: a lightweight malware detection framework on smartphones. J Netw 7(4):715–722

Zheng M, Sun M, Lui J (2013) DroidAnalytics: a signature based analytic system to collect, extract, analyze and associate android malware. http://arxiv.org/abs/1302.7212. Accessed 1st Oct 2013