Computers and Electrical Engineering 000 (2017) 1–14

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Machine learning-assisted signature and heuristic-based detection of malwares in Android devices☆

Zahoor-Ur Rehman[a], Sidra Nasim Khan[a], Khan Muhammad[b], Jong Weon Lee[b,c], Zhihan Lv[d], Sung Wook Baik[b], Peer Azmat Shah[e], Khalid Awan[e], Irfan Mehmood[f,*]

[a] Department of Computer Science, COMSATS Institute of Information Technology, Attock, Pakistan
[b] Intelligent Media Laboratory, Digital Contents Research Institute, Sejong University, Seoul, Republic of Korea
[c] Department of Software, Sejong University, Seoul, Republic of Korea
[d] School of Data Science and Software Engineering, Qingdao University, China
[e] Internet, Communication & Networks (ICNet) Research Lab, Department of Computer Science, COMSATS Institute of Information Technology, Attock, Pakistan
[f] Department of Computer Science and Engineering, Sejong University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 10 July 2017
Revised 21 November 2017
Accepted 21 November 2017
Available online xxx

Keywords:
Malware detection
Hybrid approach
Android applications
Security
Heuristic analysis

ABSTRACT

Malware detection is an important factor in the security of the smart devices. However, currently utilized signature-based methods cannot provide accurate detection of zero-day attacks and polymorphic viruses. In this context, an efficient hybrid framework is presented for detection of malware in Android Apps. The proposed framework considers both signature and heuristic-based analysis for Android Apps. We have reverse engineered the Android Apps to extract manifest files, and binaries, and employed state-of-the-art machine learning algorithms to efficiently detect malwares. For this purpose, a rigorous set of experiments are performed using various classifiers such as SVM, Decision Tree, W-J48 and KNN. It has been observed that SVM in case of binaries and KNN in case of manifest.xml files are the most suitable options in robustly detecting the malware in Android devices. The proposed framework is tested on benchmark datasets and results show improved accuracy in malware detection.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, smart devices have become main source of communication. Mobile phone has entered in market with a regular handset and now it has changed in to Smartphone with significant improvements in technology. Numbers of Smartphone users are greater than ever because of its available facilities. In past, mobile phone was only used to make phone calls and SMS messages. Recent situation has been changed and mobile phones are being used as camera, music player, Tablet PC, web browser etc. Today's mobile phones are equipped with multiple sensors with enhanced memory and processing power, thus enabling them to be used as personal computer.

A Smartphone requires applications and Operating System (OS) to facilitate users. Different operating systems are available for Smartphone's like iOS, Windows, Blackberry OS and Android. Android is the most famous among these platforms.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. S. Liu.

* Corresponding author.
E-mail addresses: irfan@sejong.ac.kr, irfanmehmood@ieee.org (I. Mehmood).

https://doi.org/10.1016/j.compeleceng.2017.11.028
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Every day, approximately 1.3 million Android devices are being activated according to Google Chairmen Erich Schmidt [1]. Android provides their users a rich media support, optimized graphic system and powerful browser. Apart from this, Android OS also provides support for 24 h GPS tracking, video camera, compass and 3D-accelerometer. It yields rich Application Program Interfaces (APIs) for location and map functions. Users can easily control or process Google map on Android devices and access location at low cost. Due to the high usage of Smartphone's, every individual user is exposed to the threat of unwanted and malicious applications. Malware authors are busy in writing malicious applications with an increase in the number of Android users. The recent research illustrated that Android Apps are repacked by malicious ELF binaries for hiding calls to external binaries. Similarly, researchers are trying to find out the best malware detection methods like memory forensic technique and secure data communication methods that can prevent Android devices.

Whenever a user wants to install an application from play store, Application is downloaded first and then asked for installation after accepting all permissions. User can't install an application without accepting all permissions required by developer (hacker). Hackers usually ask for permission through which they can access user's camera, audio, text messages and all other private information. Users are uninformed of this purpose of hackers and they accept all permissions to install application. In this way, they become victim of hacker's attack. Moreover, hackers can make changes in constant strings to attack on mobile devices, as explained in feature extraction part in Section 3.

Several techniques for detecting malware have been proposed in literature which can be divided into two broad classes: Static and Dynamic Analysis-based methods. Dynamic Analysis also known as behavioural-based analysis collects information from the OS at runtime such as system calls, network access and files and memory modifications. Hybrid apps consist of both native apps, and web apps. Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app.

Often, companies build hybrid apps as wrappers for an existing web page. In that way, they hope to get a presence in the app store without spending significant effort for developing a different app. Hybrid apps are also popular because they allow cross platform development. Thus significantly reduce development costs: that is, the same HTML code components can be reused on different mobile operating systems. Tools such as PhoneGap and Sencha Touch allow people to design and code across platforms, using the power of HTML. However, developers rush to exploit off the shelf libraries in hybrid apps. Great new features are freely available without fully understanding, addressing, the security implications, increasing the chances of malware penetration in mobile devices.

In Static Analysis (signature-based analysis), information about the App and its expected behaviour consists of explicit and implicit observations in its binary/source code. Static Analysis methods are fast and effective, but various techniques can be used to dodge Static Analysis and thus render their ability to cope with polymorphic malware. There are number of signature and behaviour-based detection tools available on play store for detection of malicious Android applications. Recent study has shown that signature based malware detection tools works till a certain level. They become ineffective when malware authors make changes in apps. Such type of signature-based tools and anti-viruses could not provide protection to Android users.

Since Android is an open source and extensible platform, it allows to extract as many features as we would like. This enables to provide richer detection capabilities, not relying merely on the standard call records or power consumption patterns. The proposed method is novel in the context that it evaluates the ability to detect malicious activity on an Android device by employing Machine Learning algorithms using a variety of monitored features like permissions, providers, intent filters, process name and constant strings extracted from Android Apps. The proposed malware detection technique can also be used on diverse environments like BlackBerry, iOS etc. This work aims to find solution for following challenges:

- How to develop a malware detection system that can adapt to any kind of malware?
- How to detect malware before actual installation?
- How to scrutinize hybrid mobile apps for possible malware threat?
- How to warn Android users about malware after a download?
- Why extracting combined features of Android Apps is better way to detect malware than signature based and behaviour-based techniques?

Selection of good features from Android applications and their combination can lead to a robust malware detection system. Most of the malware detection techniques with dynamic analysis detect malware after installation of an Android App which can affect devices. To install an application, user has to allow all malicious permissions. It is not a secure way that a malware detection technique identifies malware after a device has been affected. We performed static analysis in the proposed malware detection technique to detect malware after downloading application. In this way, security of Smartphone does not compromise. When a user downloads an App from play store and identified malicious by the proposed malware detection technique, Apps will be prompted by detection system and user will be informed about malicious app before installation.

Contributions of this work are:

- A generic malware detection framework that is adaptive to different types of malware.
- Real time detection of malware using combination of string analysis and permissions.
- Performance evaluation methods to calculate precision and accuracy of malware detection in Android Apps.

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

As we have used hybrid approach that is both static and dynamic. Static approach will overcome the drawbacks of dynamic approach, while dynamic approach will cover the deficiencies of dynamic approach. So, all kind of malwares can be detected using the proposed approach, because the hybrid approach is used in our model. In this way, our proposed generic malware detection approach can detect different types of malware.

Before installing application, our proposed method will compare constant strings of downloaded Android Package Kit (APK) with constant strings of malware applications. The APK contains all components of an Android application and used for distribution and installation of mobile applications. If constant strings of malware application and downloaded APK do not match, user will be informed that application is malicious. Secondly, our extracted malicious keywords will be compared with manifest.xml file of application to check weather application is malware or legitimate. All these processes are executed before installation of APK file. In this manner, the proposed system can detect malware before installing an application.

The remainder of this paper is organized as follows. Section 2 summarizes related work, and Section 3 introduces the proposed framework. Section 4 describes the evaluation scheme and experimental setup, finally Section 5 concludes the paper.

2. Literature review

Researchers have presented various methods for the detection of malware in Android OS. In this section, an overview of the existing malware detection frameworks is provided.

2.1. Detection of malicious applications with Static Analysis

Static analysis is an approach in which malware behaviour can be detected through source code, serial number of developer and other Meta information. Malware detection approach considering source code and serial number of developer's certificate to track intruder has been presented. While distributing an application on play store, developer signs a certificate. Serial number is assigned to developer certificate and developer can be traced out by searching for serial number.

Static assurance analysis of Android applications has been implemented depending on user intentions. In static assurance analysis of application, percentages of dangerous function calls are calculated that rely on performance of user dealings. Critical function calls value is considered assurance score. Furthermore, static assurance analysis can be applied on applications of other operating systems. In [2], a method proposed by using static analysis to compare application with previously seen malwares. Similarity measure had used for comparison of application with previously known malware. In [3], static analysis approach presented by extracting strings from disassembled Android applications and then VSM (Vector Space Model) is used to represent strings of Android application like vector in space. Afterward Euclidean distance, Cosine Similarity and Manhattan distance are applied on strings to check malwares in applications. Drebin [4] is another tool to detect malware more accurately with static analysis. When users install application, it alerts users with explanation that scanned application is malicious.

A novel malware detection method using Bayesian classification is developed [5]. Bayesian classifier method uses static analysis by reverse engineering Android applications. Features are extracted by APK (Android Package Kit) tools and then Bayesian classifier is implemented on them to detect malicious code. Another malware detection static analysis approach is presented using inter-component communication taint analysis tool and inter-component communication to detect information leaks. An approach to find hijacking susceptibilities in Android applications by chase stains between sensitive sources and outwardly accessible interfaces is proposed. Leak Miner and Android Leaks utter the power to control the Android lifecycle together with call back ways; however, both tools are not context-sensitive which prevents the exact analysis of the several sensible eventualities.

Flowdriod [6] is static taint analyses tool that analyze byte code and configuration files of application in Android platform to detect privacy leaks. Flowdroid provided efficacy and high precision but cannot perform inter-component communication. To overcome this problem, many other malware detection approaches have been proposed but there are still some major deficiencies in performing static analysis completely. Such approaches are unable to carry out deep static analysis failing to find solution of reflective methods calls. SAM (Static Analysis Module) [7] has been presented for mobile application verification cluster to analyze security of third party applications on different play stores. Android platform is used for implementation and testing SAM.

AndroSimilar [8] is a tool to detect malwares by extracting features from Android APK and then signature is generated from malware database to match with unknown application signatures. If signature does match with unknown Android application, it is considered as malicious. DroidAnalytics [9] is static analysis tool to detect malware with signature generation at opcode level. Signature generation is used to find out obfuscated code and repacked malware in applications. For detection of repacked malware, similarity core is used in given method. DroidAnalytics is useful to detect obfuscation methods e.g. string encryption and method renaming.

2.2. Detection of malicious applications with dynamic analysis

Dynamic analysis is process of executing applications to detect vulnerabilities. Dynamic analysis is more complicated as compared to static analysis. It is not easy for an attacker to stop dynamic analysis, as dynamic analysis is deployed out-

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

wardly. Researchers tried to developed dynamic analysis techniques to overcome problems of static analysis methods**.** Number of dynamic analysis practices for privacy leaks in Android application has been presented. Some of dynamic analysis methods are presented in this section. McClurg has presented java-based dynamic analysis approach [10] to track applications in case of privacy leaks. Given approach has taint propagation functionality and generates alerts when sensitive information leaks in Android environment.

Malicious behaviour of applications has been detected by applying both dynamic and static analysis. First applications are analyzed through static analysis without installing them and then dynamic analysis is deployed by executing applications. Thus, static and dynamic analysis called hybrid analysis method is implemented to detect susceptibilities in applications. Some efficient tools are being used to find malicious applications on Android operating system but they do not support API call tracking while applying dynamic analysis. Similar malware detection approaches like Android Bouncer provides security to users by analyzing applications on Android market. When an Android application is uploaded on play store, Bouncer starts analyzing it and searching for malware, Trojans and spywares. Analysis has been performed on Google cloud to check malicious actions of an application for user's security. DroidScope [11] has ability to carry out analysis of java and native components of application to search out malicious patterns. Apps Playground [12] is dynamic analysis method that automatically observes and detects malwares in Android applications by using exploration procedures.

Mobile-Sandbox [13] is hybrid approach combining both static and dynamic analysis to record malicious patterns of applications. First static analysis is done for analysis of source code and to find out malicious permissions. Later, dynamic analysis is performed by executing application to record all actions even generated from native API calls. ANDRUBIS [14] is automatic approach combining both dynamic and static analysis that logs events by executing java code in Dalvik Virtual Machine and native code at system level. Often attackers use anti-detection techniques to hide maliciousness of applications from malware detection tools by detecting virtual environment/emulated environment. If virtual environment/sandbox is detected they stop execution of malicious behaviour and application is considered legitimate or good ware. To deal with such problem, a dynamic malware detection method [15] has been presented that is flexible against anti-detection techniques of malware attackers.

2.3. Machine learning-based malware detection approaches

Concept of machine learning is to let the algorithms learn by itself the required parameter from data in order to make finest predictions. There are various machine learning approaches available for data security and detection of malicious applications [16,17]. New machine learning framework has proposed for detection of malware that extracts several features from applications and train SVM in offline mode to avoid high computing resources [18]. Andromaly [19] is a behaviour based machine learning approach to detect malware from Android applications. It is host-based malware detection approach that monitors features and events from application running on mobile device on the basis of known malware samples and then applies classifiers for the detection of malicious applications. Drebin [4] is another approach that collects number of features from Android applications and stores them as a pattern recognition. It takes decision on the basis of saved patterns, whether application is legitimate or malicious. Another machine learning approach [20] has been proposed for malware detection from Android applications. It is permission based malware detection approach that gathered a variety of permissions and actions from applications first and then machine learning classifiers were applied on collected features.

PMDS (Permission based Malware Detection System) [21] has dug out permissions from Android applications and applied machine learning classifiers to identify unknown malwares. PMDS is behaviour based approach and achieved 94% accuracy detection of malicious apps with 1.52 to 3.93% false positive alarm. A parallel machine learning approach [22] was proposed by extracting three features (1) permission related features (2) command related features (3) and API calls related features. After extraction of features, classifiers were applied to detect malware. Machine learning ensemble method [23] Troika was used as a classifier after getting permissions and API calls from apps. Troika is a single classifier that performs as a multiple classifier to improve accuracy results.

Since, Android is an open source and extensible platform it allows to extract as many features as we would like. This enables to provide richer detection capabilities, not relying merely on the standard call records. The proposed method is robust and it employs light weight Machine Learning algorithms to ensure scalability and maintains low resource consumption of the detection process.

3. Methodology

To overcome the problems of static and dynamic malware detection techniques proposed in literature and summarized in Section II, this section describes the details of a new hybrid malware detection model for Android applications. The proposed model uses both static and dynamic analysis for malware detection. Malwares that can't be detected in static approach, dynamic approach will detect them and the malwares that can't be detected in dynamic approach, will be detected by the static approach. So, all kind of malwares will be detected because of hybrid approach used in our model. In addition to this, our proposed method compares the constant strings of downloaded APK file with the constant strings of malware applications and with manifest.xml file of application to check weather application is malware or legitimate. All these processes are executed before the installation of APK file. In this way, the proposed system can detect malware before installing an application.

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

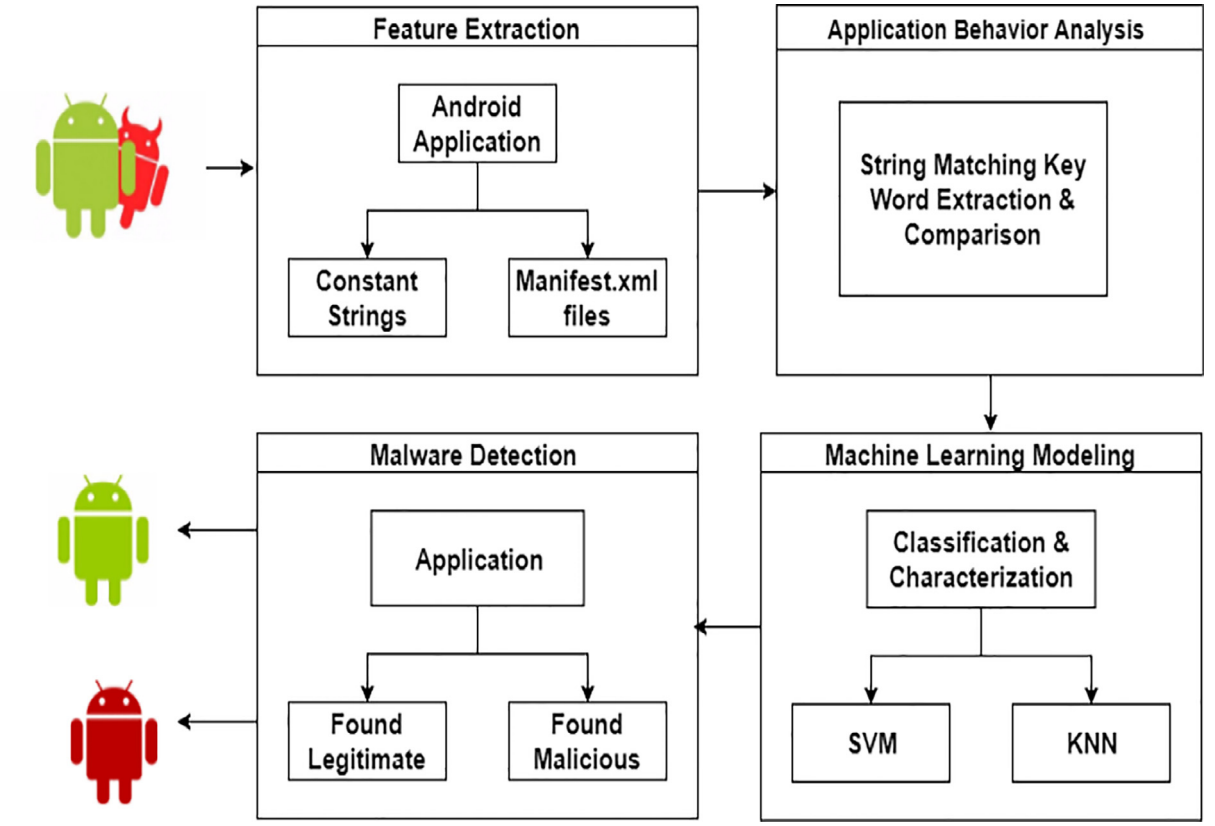

Fig. 1. Overview of the proposed framework for Android malware detection.

The proposed method consists of three steps as shown in Figs. 1 and 2. In the first step, constant strings of Android applications are extracted and Android manifest.xml files of applications are explored. In the second step, constant strings and keywords are separated from Android manifest.xml file of malicious and legitimate applications. Third step involves classification of malware and legitimate applications using keywords and constant strings as input features. After classification and characterization, malicious and legitimate applications are identified with labels. Subsequent sections provide details of each module.

3.1. Features extraction

To build up an efficient Android malware detection model, it is highly desirable to collect robust and most representative features such as user permissions, providers and receiver's information, intent filters, process name and binaries from under analysis applications. In the proposed framework, all the mentioned features are extracted via reverse engineering, breaking down its application APK in to simple java code and after modification it is again converted in to an APK file. In this context, Easy APK Disassembler is adopted to reverse engineering the applications. After reverse engineering the Android application, features are extracted that are constant strings from binaries and permissions, providers, receivers, intent filters, process name from Android manifest.xml files.

Constant strings are actually binaries of Android applications that exist in folder after reverse engineering of an application. Attackers can make change in constant strings to attack a device by reverse engineering an application. As we get source code of an application after reverse engineering process, hackers make changes in constant strings, repack that application and upload on play stores. If an application has constant string [const-string v1, "Rooted Device"], malware attackers can change it to [const-string v1,"Device Not Rooted"]. Similarly, they can attempt multiple attacks by making changes in constant strings of Android applications. While every Android application has one manifest.xml file in which all permissions from users are requested. Users can't install application without accepting all requested permissions.

Along with constant strings, keywords (manifest feature) are also extracted from applications. For this purpose, list of keywords is created considering the information from existing malicious and legitimate application as shown in Table 1. Table 1 contains keywords that are extracted from malicious manifest.xml files of Android applications. These extracted keywords help in detection of malicious applications. Every keyword listed in Table 1 have specific purpose in manifest.xml

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

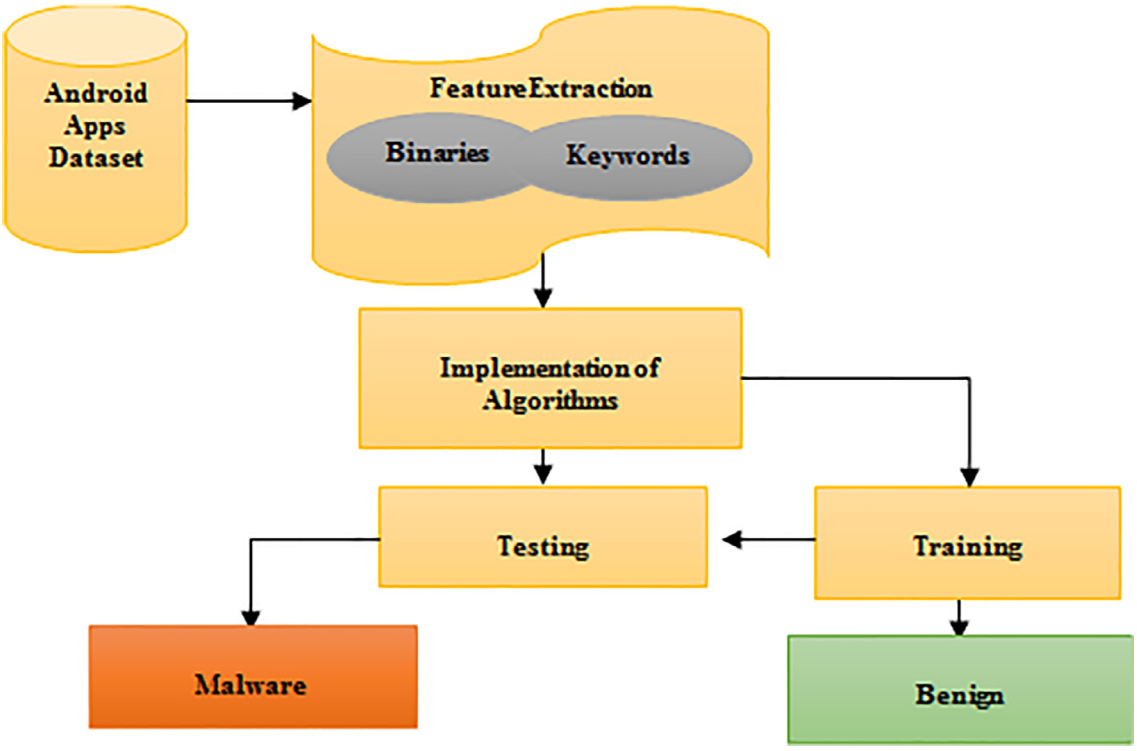

Fig. 2. Detailed view of the proposed framework for Android malware detection.

file like "Read SMS" read sent messages and received messages of Android phone user. Likewise "Read Phone State" reads phone number in phone of user.

READ_SMS, SEND_SMS, RECEIVE_SMS, WRITE_SMS and all other keywords listed in Table 1 are found frequently in malicious applications. By analyzing malicious applications, we extracted all of keywords that found frequently in manifest file of applications. After extraction of keywords, the proposed keyword structure is compared with manifest.xml files of legitimate and malicious applications. Afterwards malignancy scores of applications are calculated by automatically adjusting threshold value and applying different algorithms.

We have selected malicious keywords from malware application and designed a data structure of malicious keywords. We have malicious and legitimate applications in data set and used supervised classification. So, we analyzed all manifest.xml files of malicious applications in our data set and designed a data structure. Table 1 presents our extended keywords list that is utilized in our model. Table 2 is the list of keywords that are utilized in already developed model. We enhanced keywords than previously developed model as shown in Table 1.

The keywords in Table 1 having **bold** font are keywords that we enhanced for more accurate detection as compared to the previously developed malware detection model. Purpose of two tables "Tables 1" and "2" is to explain that how many keywords were used in previously developed model and how many keywords have been utilized in our proposed model. Actually, we are comparing our keywords with the already developed malware detection models.

In the proposed method more keywords are extracted than previously developed model [24] by making use of manifest files. We searched out more malicious keywords (bold font in Table 1) in malware manifest.xml files to improve accuracy results than already proposed malware detection technique. In previously developed model, four lists of features were utilized but we extended it to eight lists. Table 2 indicates that keywords (manifest features) were lesser in already proposed framework of malware detection. We enhanced the keywords and got high accuracy rates with better detection of malicious applications. Furthermore, the proposed machine learning model yields improved security to Android users.

3.2. App classification into benign and malignant

At first, we collected data set of Android applications and then extracted features from malware and benign applications. Binaries and manifest.xml files are included in extracted features. Constant strings, API calls, system calls are built-in binaries while permissions, intent filters, broadcast receivers, providers and process name are integrated in keywords. We randomly gathered constant strings from benign and malware apps. In case of keywords (manifest features), we collected Android manifest.xml files of Android applications by analysis malware samples. Database was created by searching services,

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Table 1
Proposed keyword list of Android applications.

Keyword list extracted from Manifest.xml Files	
(List 1)Permission	2.main
1.READ_SMS	3.two
2.SEND_SMS	4.three
3.RECEIVE_SMS	**(List 5)Provider**
4.WRITE_SMS	1.READ_SETTINGS
5.PROCESS_OUTGOING_CALLS	2.WRITE_SETTINGS
6.MOUNT_UNMOUNT_FILESYSTEMS	3.AdContentProvider
7.READ_HISTORY_BOOKMARKS	4.READ_WRITE
8.WRITE_HISTORY_BOOKMARKS	5.ChompProvider
9.READ_LOGS	**(List 6)Intent filter(scheme)**
10.INSTALL_PACKAGES	1.sms
11.MODIFY_PHONE_STATE	2.smsto
12.READ_CONTACTS	3.chomp
13.ACCESS_WIFI_STATE	4.file
14.DISABLE_KEYGUARD	5.content
15.CHANGE_WIFI_STATE	**(List 7)Receiver**
16.SYSTEM_ALERT_WINDOW	1.OnBootReceiver
17.CHANGE_CONFIGURATION	2.AutorunBroadcastReceiver
(List 2)Intent-filter(action)	3.remote
1.BOOT_COMPLETED	4.SMSReceiver
2.SMS_RECEIVED	5.SecurityReceiver
3.CONNECTIVITY_CHANGE	6.RepeatingAlarmService
4.USER-PRESENT	7.Checker
5.PHONE-STATE	8.AdNotification
6.NEW_OUTGOING_CALL	9.GCMBroadcastReceiver
7.UNINSTALL_SHORTCUT	10.SEND
8.INSTALL_SHORTCUT	11.MessageReceiver
9.left_up	12.OnLogAlarmReceiver
10.right_up	13.DateTimeReceiver
11.left_down	14.AutoAnswerReceiver
12.right.down	15.SmsSendingReceiver
13.SIG_STAR	16.WidgetEventReceiver
14.VIEW(benign keyword)	17.ActionReceiver
15.MAIN	18.SecurityReceiver
16.CALL	19.BIND_DEVICE_ADMIN
(List 3)Intent_filter(category)	
	(List 8)Intent_filter(priority)
1.HOME	
2.BROWSABLE(benign keyword)	1.1000
3.LAUNCHER	2.2147483647
(List 4)Process name	3.100
1.remote2	

Table 2
Previously used keyword list of Android applications.

(List 1) Permission	9. left_up
1. READ_SMS	10. right_up
2. SEND_SMS	11. left_down
3. RECEIVE_SMS	12. right_down
4. WRITE_SMS	13. SIG_STR
5. PROCESS_OUTGOING_CALLS	14. VIEW (benign keyword)
6. MOUNT_UNMOUNT_FILESYSTEMS	**(List 3) Intent-filter (category)**
7. READ_HISTORY_BOOKMARKS	1. HOME
8. WRITE_HISTORY_BOOKMARKS	2. BROWSABLE (benign keyword)
9. READ_LOGS	**(List 4) Process name**
10. INSTALL_PACKAGES	1. Remote2
11. MODIFY_PHONE_STATE	2. main
(List 2) Intent-filter (action)	3. Two
1. BOOT_COMPLETED	4. Three
2. SMS_RECEIVED	
3. CONNECTIVITY_CHANGE	
4. USER_PRESENT	
5. PHONE_STATE	
6. NEW_OUTGOING_CALL	
7. UNINSTALL_SHORTCUT	
8. INSTALL_SHORTCUT	

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

broadcast, activities and broadcast receivers from Android manifest.xml files of applications. We created data structure by analyzing frequent malicious keywords exist in malware samples.

Various parameters and algorithms were applied to get the perfection of malware detection model. Like Constant Similarity, Manhattan Distance, Inner Product Similarity, Kernel Euclidean Distance were applied as a data to similarity measure. Cosine Similarity suits well with the proposed malware detection model in achieving high percentage of accuracy. Apart from this, we have utilized automatic sampling type; parallelize training and testing with 10-fold cross validation technique. Automatic sampling type usually use stratified sampling but if stratified samples are not available in examples set, it uses shuffled samples instead of stratified samples. We set out operator for finding threshold automatically and then applied threshold that was suitable for proposed model. After setting out parameter and implementation of best fit algorithms, we performed training and then testing to get overall results. At end, proposed malware detection model indicates that sample is malware or benign. For classification and implementation of the algorithm, we used 10-fold cross validation technique to validate performance of the proposed. In 10-fold cross validation technique, data is divided in to 10 parts for training and testing. One part is used for testing and remaining k-1 parts are used for training. There are three ways to split data, training, testing and validation. Validation is performed after testing of data to check out over fitting issues. To avoid over fitting problems by implementation of algorithms, validation is performed. Basic advantage of over fitting is that every part of data has used for both training and testing.

4. Experimental results, and discussion

The proposed framework is tested on small data sets, M0DROID [25]. This dataset contains different type of android applications like: games, selfie camera, torch apps, weather apps, Map apps, music, health and many others. We tested our framework with all existing types of malware families; some of them are Plankton, DroidKungFu, GinMaster, FakeInstaller, Opfake, BaseBridge, Nisev, Adrd, Kmin, Geinimi, DroidDream, Imlog, Nandrobox, SmForw, Plankton, FakeRun and many more. Our hybrid approach detected all types of malwares listed above.

First, we applied algorithms with suitable parameters separately on Android permissions, intent filters, receivers, providers and broad cost receivers that we have called keywords in the proposed work. We prepared our own data structure with extracted keywords from manifest files of malicious apps and compared it with manifest.xml files of legitimate and malicious samples. Similarly, we extracted binaries from Android apps and implemented various parameters and algorithms to check out accuracy results. At the end, we combined both keywords (manifest features) and binaries (constant strings) of applications, compared with the proposed malicious and legitimate samples existing in data sets, achieved secure malware detection technique with highest accuracy results.

4.1. Dataset

We used M0DROID [25] data sets to build up model for malware detection in Android applications. Our data set M0DRIOD contains 204 legitimate and 197 malicious Android applications. To train malware detection model, we utilized 137 samples out of 204 samples of legitimate applications and 137 samples of malicious application when we compared manifest.xml files with our prepared data structure. For testing of model, 67 legitimate samples and 60 malicious samples were utilized. In case of combined samples, we have used 274 malicious samples in which 137 binary files and 137 manifest.xml files were compared with our self-defined malicious data structure (137).

4.2. Experimental setup

Various experiments are performed to measure the performance and effectiveness of the proposed machine learning model. Designed machine learning model to detect malwares is shown in Fig. 3. In addition, we have used CORE I5 LATITUDE/E5410 with 4 GB random access memory for training and testing of proposed model. Window 8, 64-bit operating system, x64 based processor was used to generate presented malware detection model. Execution time of the proposed model depends of on algorithms and it takes 0.1 MB memory but execution time varies in some cases.

4.3. Quantitative and qualitative analysis

The malware detection model is shown in Fig. 3, in which we first compared only malicious binaries with legitimate binaries that were extracted from APK files exists in data sets. Then, we applied cosine similarity as data similarity measure and performed 10-fold cross validation on binaries only. Secondly, we compared only manifest files of legitimate and malicious applications separately with our prepared data structure as shown in Table 1 (Data Structure) and performed cosine similarity before validation. At end, we combined binaries (constant strings) and manifest.xml files (manifest features) files and compared them with our prepared data structure and applied similar steps like data to similarity and validation.

We used Cosine similarity [8] to calculate distance between two vectors in the proposed machine learning technique. Usually Cosine similarity is used to calculate similarity between two vectors by finding cosine angle. Where “v” and “u” are vectors from point “x” and “y”, v.u is inner point and ||v||.||u|| is cross product of “u” and “v”. In Table 3, it is indicated that similarity exists between malware and legitimate samples of Android applications. Only similarity between few numbers of

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

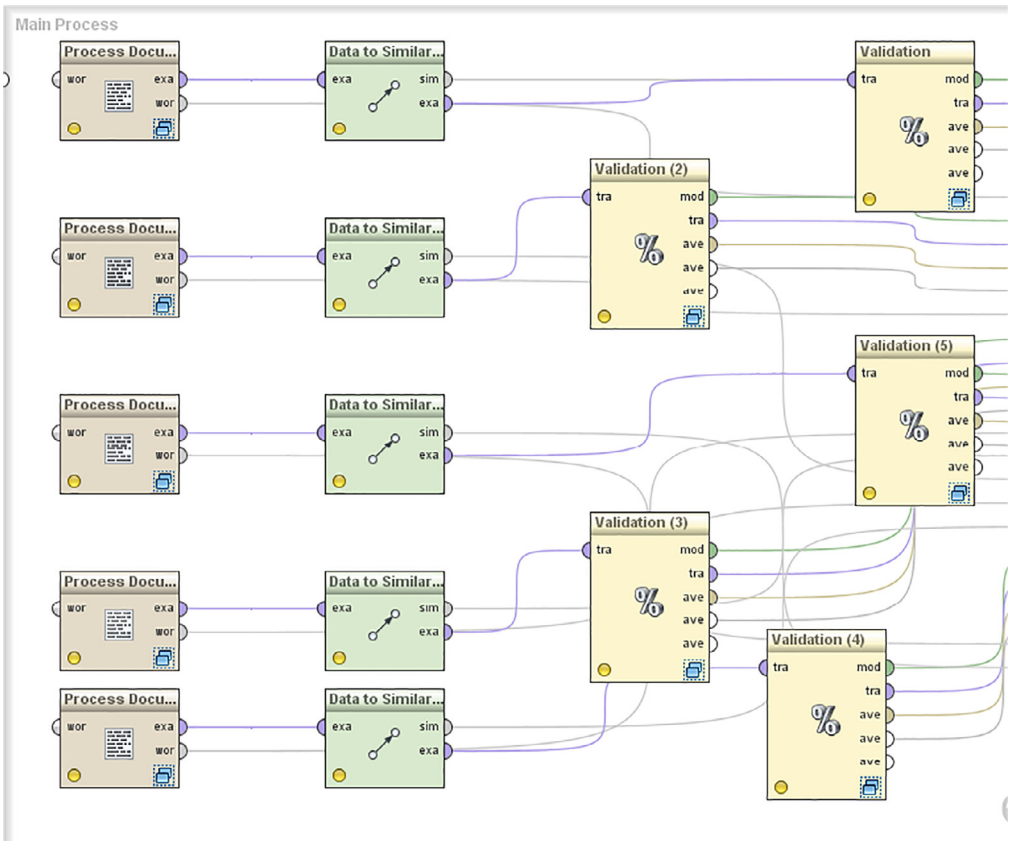

Fig. 3. Proposed machine learning malware detection model.

Table 3
Similarity between malware and legitimate samples.

First doc	Second doc	Similarity value
524.0	525.0	0.709
524.0	526.0	0.876
524.0	527.0	0.831
524.0	528.0	0.948
524.0	529.0	0.795
524.0	530.0	0.810
524.0	531.0	0.492
525.0	526.0	0.796
525.0	527.0	0.656
525.0	528.0	0.666
525.0	529.0	0.628
525.0	530.0	0.606
525.0	530.0	0.635

documents is shown in Table 3 just to explain results of the proposed model. We got similarity between documents in form of “0”and “1”. In Table 3, we found similarity between “0” and “1”. It means, two documents are similar to some extent. For instance, document “524” and “525” have similarity of “0.709” near to one.

In Fig. 4, “similarity value” is the similarity between two documents and *“words frequency”* indicates frequency of word in documents. “Similarity value” is value of similar words found in two documents that how much similar words exist in two different documents as shown in Table 3. Likewise, word frequency is related to how frequently words exist in documents over all when comparing one document to other. Here, in histogram words frequency is too high because counting of similar word starts when first document compares with second. Again, first document compared with third document and then with fourth, fifth document and so on. Now, second document starts comparison with second document, third, fourth and continue comparison till the last document. Similarly, other documents that we provided as input start comparing each

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

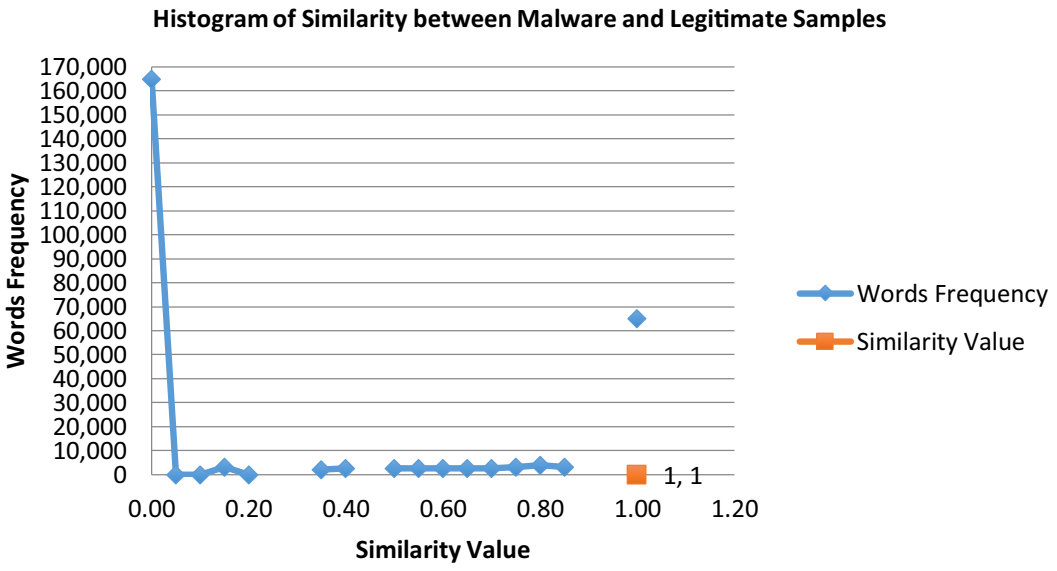

Fig. 4. Histogram of similarity between malware and legitimate samples.

Table 4
Acquired results while using binaries as training attributes.

Algorithms	TPR	FPR	AUC	Accuracy	Standard deviation	Time
Linear discriminant analysis	0	53.84	0.50	46.15	0.00	30:16
Random tree	0	53.84	0.50	46.15	0.00	3:01
Decision stump	0	53.84	0.73	46.15	0.00	5:34
W-J48	93.54	48.47	0.87	56.54	12.41	2:46
KNN	84.70	38.85	0.50	68.85	15.90	3 sec
W-J48 graft	86.04	47.46	0.86	58.08	12.92	2:17
Support vector machine (SVM)	80.23	6.45	0.88	85.00	4.37	1:25

other and show similarity in the form of “1” and “0”. Similar words are continuously counted in this process that’s why histogram is indicating highest word frequency of “160,000”.

Different algorithms were implemented on dataset in order to achieve high accuracy. Some algorithms failed to get good accuracy and end up with worst results. For example, Linear Discriminant Analysis obtained worst accuracy of 46.15%. Similarly, Random tree achieved 46.15% accuracy results. On the other side, SVM was successful in acquiring best accuracy rates of 85.00%.First, we only used binaries (constant strings) of applications and we got better performance as compared to already developed model of malware detection. We got 85% accuracy with SVM with standard deviation of 4.37 shown in Table 4.

Standard deviation can affect accuracy that’s why algorithm with low standard deviation considered best. The SVM got low standard deviation as compared to W-J48, KNN and W-J48 graft. While, Linear Discriminant Analysis, Random Tree and Decision Stump gives out 0.00 standard deviation but their accuracy is too low. Execution time of SVM is also less if we compare it with other utilized algorithms. KNN has very low execution time that is 3 s but accuracy is low in that case. Overall, SVM suits best with binaries as compared to other algorithms. Previously developed model [8] used VSM (Vector Space Model) for representation of documents and got 83.51 accuracy results. The proposed model improved accuracy and achieved accuracy rate of 85%.

4.4. Comparative analysis with state-of-the-art methods

We achieved improved detection rates as compared to already developed model. We got 85% accuracy rates with true positive rate of 80.23% and false positive rate of 6.45%.Previously developed model [8] got 83.51% accuracy rates with false positive rate of 27% and true positive rate of 94%. The proposed model was successful to achieve high accuracy with low false alarming rates as compared to already developed model as shown in Table 5. We also used extracted keywords from manifest.xml files of applications to get more accurate results and improved accuracy. First, we compared only malicious manifest.xml files with our prepared data structure of extracted keywords. Later, we also compared legitimate manifest with data structure to check accuracy of machine learning tool (Rapid Miner). Data structure was prepared by extracting keywords from manifest.xml files (Permissions, Providers, Receivers, Intent Filters, and Process Name). We deployed different algorithms on malicious manifest files to check accuracy results. Worst accuracy results were achieved by applying Linear

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Table 5
Comparison of proposed model with previously developed model (constant strings).

	TPR	FPR	Accuracy
Previously developed model	94%	27%	83.51%
Proposed model	80.23%	6.45%	85.00%

Table 6
Results by comparing malicious manifest with data structure.

Algorithms	TPR	FPR	AUC	Accuracy	Standard deviation	Execution time
Linear discriminant analysis	83.33	49.59	0.50	50.73	1.67	30:16
Random tree	1	39.11	0.50	67.91	10.53	3:01
Decision stump (Tree)	1	32.17	0.50	76.31	7.12	5:34
W-J48	1	6.16	0.50	96.77	3.73	2:46
KNN	1	2.14	0.50	98.90	1.68	3s
W-J48 graft	1	6.16	0.50	96.77	3.73	2:17
SVM	1	18.45	1	88.76	6.50	1:25

Table 7
Acquired results while using legitimate manifest with data structure.

Algorithms	TPR	FPR	AUC	Accuracy	Standard deviation	Execution time
Linear discriminant analysis	53.57	49.59	0.50	50.73	1.67	30:16
Random tree	1	39.11	0.50	67.78	9.40	3:01
Decision stump (Tree)	1	29.74	0.50	78.90	7.25	5:34
W-J48	1	9.86	0.50	94.56	4.34	2:46
KNN	1	18.45	0.50	88.68	7.07	3s
W-J48 graft	1	9.86	0.50	94.56	4.34	2:17
Support vector machine (SVM)	1	24.30	0.99	83.96	10.46	1:25

Table 8
Comparison of developed model with previously proposed model.

	TPR	FPR	Accuracy
Previously developed model	87.5%	12.5%	90%
Proposed model	100%	0.38%	99.81%

Discriminant Analysis algorithm while best accuracy was achieved by KNN with standard deviation of 1.68, as shown in Table 6.

KNN's execution time was recorded 3 s only while SVM, Weka-J48, Weka-J48 graft was also executed in less time. FPR ratio of KNN is 2.14 while TPR is 1. Here 1% does not mean over fitting because we provided data structure instead of samples for comparison. That is the reason behind getting TPR of 1%. Linear Discriminant Analysis could not identify data structure properly that's why it TPR is 83.33%. While other algorithms identify our provided data structure and do not take them as samples. We compared legitimate manifest with data structure to check that whether our utilized tool can accurately detect legitimate samples or not. For this purpose, same parameters were used as in case of comparing malicious samples with data structure. In results, we have seen accuracy and perfection of our tool (Rapid Miner).

Achieved results are shown in Table 7. The proposed model gives high accuracy rates as compared to previously developed model. We got very low alarming rate of 0.38% while previously proposed model have more alarming rate of 12.5% as shown in Table 8. TPR of 1 is achieved by all utilized algorithms except Linear Discriminant Analysis. Reason behind TPR of 1 is not over fitting but reason is our designed data structure that we provided as input samples to compare with malware samples.

The Linear Discriminant Analysis considered our provided data structure as sample. Previously developed model had achieved 90% accuracy by using lesser manifest features and W-J48. We implemented W-J48 on more extracted features and got 96.77% accuracy with low alarming rates of 6.16%. To improve the accuracy, we implemented KNN and got best accuracy of 98.81. We again combined binaries and manifest files and checked out performance of model. We got more improved accuracy of 99.81% while using combined features as compared to utilizing separate features. For instance, we first used only binaries and then used only manifest files in previous experiments. Here, we can observe that the proposed model is effective to the some extent.

We wanted to improve the accuracy results for more accurate detection of malwares. For this purpose, we used combination of malicious manifest and binaries. We got highest accuracy results by making use of combined features and achieved 99.81% accuracy with lowest standard deviation. We implemented KNN algorithm on combined features to reach high accu-

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Table 9
Acquired results while using combination of malicious manifest and binaries.

Algorithms	TPR	FPR	AUC	Accuracy	Standard deviation	Execution time
Linear discriminant analysis	0	0.51	0.50	48.40	0.85	30:16
Random tree	1	32.89	0.50	76.30	12.05	3:01
Decision stump (Tree)	1	31.09	0.50	78.15	3.63	5:34
W-J48	1	4.81	0.50	97.55	2.67	2:46
KNN	1	0.38	0.50	99.81	0.56	3s
W-J48 graft	1	3.74	0.50	98.11	2.23	2:17
SVM	1	7.88	1	95.86	2.77	1:25

Table 10
Acquired results while using legitimate manifest and binaries.

Algorithms	TPR	FPR	AUC	Accuracy	Standard deviation	Execution time
Linear discriminant analysis	0.66	0	0.50	33.08	1.30	30:16
Random tree	1	55.94	0.50	58.10	17.89	3:01
Decision stump (Tree)	1	55.66	0.50	58.47	4.22	5:34
W-J48	1	5.51	0.50	98.06	2.39	2:46
KNN	1	13.83	0.50	94.67	2.62	3s
W-J48 graft	1	5.51	0.50	98.06	2.39	2:17
SVM	1	50.89	1	65.69	4.81	1:25

racy results of 99.81% as shown in Table 9. Here, we got desired results. The proposed model obtained highest accuracy as compared to all previously developed models according to our knowledge. As we already compared legitimate manifest.xml files with data structure to check accurate detection of our used tool to design model. Again, we will use combination of legitimate manifest and binaries to check out accuracy of tool and achieved improved accuracy as shown in Table 10.

Previously developed techniques are being used for malware detection but some of them are using only permissions and some of them are using only API call etc. Such techniques are good to be some extent but when malware writers' start making changes in code, these techniques got failed at this point. The proposed malware detection technique overcomes such troubles because we tried to utilize almost every feature of application like in keywords, permissions, providers, receivers, intent filters were extracted.

Likewise, mixture of simple and constant strings in binaries was extracted. In this prospective, the proposed malware detection technique gives full protection to Android users. Another advantage of the proposed technique is that it can be used in diverse environments like iOS, Blackberry because applications of all OS have permissions and binaries.

5. Conclusion

In last two decades various malware detection approaches with static and dynamic analysis have been developed, however both have some pros and cons. In this work, a hybrid approach using both static and dynamic analysis methods is presented that improved the overall accuracy of malware detection, while supplementing the drawbacks of static and dynamic analysis methods. Two types of features are extracted from various android Apps. These features are broadly classified into three categories: users' permissions, keywords from manifest.xml files and strings from other files of applications. These features were used as input to different machine learning classifiers to detect malware. In the first step, features extracted from only binaries were used to classify Apps. Then, keywords and features from manifest.xml files of Apps were used to detect malware. Finally, all the extracted features were fused and combined into a single feature vector to obtain more accurate results. It is observed that by features fusion malware detection accuracy has increased significantly.

We have incorporated various classifiers in order to classify apps as either legitimate or malicious. The lowest accuracy was achieved by Linear Discriminant Analysis (46.15%), and similar is with Random tree (46.15%). The highest accuracy was achieved with the SVM and it was equal to 85.5% with standard deviation of 4.37. Thus, SVM suits best with binaries as compared to other algorithms and the proposed model improves the accuracy to 85.51%, which was 83.5% when used with Vector Space Model. It is recommended to consider implementing SVM for binary classification. The proposed framework can be extended to update the designed data structure of malicious keywords for detection of newly created malwares.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00312) supervised by the IITP (Institute for Information & communications Technology Promotion).

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Competing interests

The authors declare that they have no competing and financial interests.

References

[1] Russello G, Jimenez AB, Naderi H, van der Mark W. Firedroid: hardening security in almost-stock android. In: Proceedings of the 29th annual computer security applications conference. ACM; 2013. p. 319–28.
[2] Jacob G, Comparetti PM, Neugschwandtner M, Kruegel C, Vigna G. A static, packer-agnostic filter to detect similar malware samples. In: *Detection of intrusions and malware, and vulnerability assessment*. Springer; 2012. p. 102–22.
[3] Sanz B, Santos I, Ugarte-Pedrero X, Laorden C, Nieves J, Bringas PG. Anomaly detection using string analysis for android malware detection. In: International joint conference SOCO'13-CISIS'13-ICEUTE'13. Springer; 2014. p. 469–78.
[4] Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K. DREBIN: effective and explainable detection of android malware in your pocket. Symposium on network and distributed system security (NDSS); 2014.
[5] Yerima SY, Sezer S, McWilliams G, Muttik I. A new android malware detection approach using bayesian classification. In: *2013 IEEE 27th international conference* on *advanced information networking* and. *applications*. IEEE; 2013. p. 121–8.
[6] Arzt S, et al. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In: *ACM SIGPLAN notices*, Vol. 49. ACM; 2014. p. 259–69.
[7] Armando A, Chiarelli G, Costa G, De Maglie G, Mammoliti R, Merlo A. Mobile app security analysis with the MAVeriC static analysis module. JoWUA 2014;5(4):103–19.
[8] Faruki P, Ganmoor V, Laxmi V, Gaur MS, Bharmal A. Androsimilar: robust statistical feature signature for android malware detection. In: Proceedings of the 6th international conference on security of information and networks. ACM; 2013. p. 152–9.
[9] Zheng M, Sun M, Lui J. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware. In: 12th IEEE international conference on trust, security and privacy in computing and communications. IEEE; 2013. p. 163–71.
[10] McClurg J, Friedman J, Ng W. Android privacy leak detection via dynamic taint analysis. *Electric. Eng. Comput. Sci.* 2013;450.
[11] Yan LK, Yin H. Droidscope: seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis. In: *21st USENIX security symposium (USENIX security 12)*; 2012. p. 569–84.
[12] Rastogi V, Chen Y, Enck W. AppsPlayground: automatic security analysis of smartphone applications. In: *Proceedings of the third ACM conference on data and application security and privacy*. ACM; 2013. p. 209–20.
[13] Spreitzenbarth M, Freiling F, Echtler F, Schreck T, Hoffmann J. Mobile-sandbox: having a deeper look into android applications. In: *Proceedings of the 28th annual ACM symposium on applied computing*. ACM; 2013. p. 1808–15.
[14] Weichselbaum L, Neugschwandtner M, Lindorfer M, Fratantonio Y, van der Veen V, Platzer C. Andrubis. Andrubis, 1. Vienna University of Technology; 2014. Tech. Rep. TRISECLAB-0414.
[15] Gajrani J, Sarswat J, Tripathi M, Laxmi V, Gaur M, Conti M. A robust dynamic analysis system preventing SandBox detection by Android malware. In: Proceedings of the 8th international conference on security of information and networks. ACM; 2015. p. 290–5.
[16] Sajjad M, Muhammad K, Baik SW, Rho S, Jan Z, Yeo S. Mobile-cloud assisted framework for selective encryption of medical images with steganography for resource-constrained devices. Multimed Tools Appl 2017;76(3):3519–36.
[17] Muhammad K, Sajjad M, Mehmood I, Rho S, Baik SW. A novel magic LSB substitution method (M-LSB-SM) using multi-level encryption and achromatic component of an image. Multimed Tools Appl 2016;75(22):14867–93.
[18] Sahs J, Khan L. A machine learning approach to android malware detection. In: Intelligence and security informatics conference (EISIC). IEEE; 2012. p. 141–7.
[19] Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y. "Andromaly": a behavioral malware detection framework for android devices. J Intell Inf Syst 2012;38(1):161–90.
[20] Aung Z, Zaw W. Permission-based android malware detection. Int J Sci Technol Res 2013;2(3):228–34.
[21] Rovelli P, Vigfússon Ý. PMDS: permission-based malware detection system. In: International conference on information systems security. Springer; 2014. p. 338–57.
[22] Yerima SY, Sezer S, Muttik I. Android malware detection using parallel machine learning classifiers. In: Eighth international conference on next generation mobile apps, services and technologies. IEEE; 2014. p. 37–42.
[23] Peiravian N, Zhu X. Machine learning for android malware detection using permission and API Calls. In: Proceedings of the 25th international conference on tools with artificial intelligence; 2013. p. 300–5.
[24] Sato R, Chiba D, Goto S. Detecting Android malware by analyzing manifest files. In: Proceedings of the Asia-Pacific advanced network, vol. 36; 2013. p. 23–31.
[25] Damshenas M, Dehghantanha A, Choo K-KR, Mahmud R. M0droid: an android behavioral-based malware detection model. J Inf Privacy Security 2015;11(3):141–57.

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028

Zahoor-ur-Rehman has experience both in academia and research. He has received his educational and academic training at University of Peshawar, Foundation University Islamabad and UET Lahore, Pakistan. He joined COMSATS Institute of Information Technology as assistant professor in the early 2015. Along with teaching responsibilities, he is an active researcher and reviewers of various conferences and reputed journals.

Sidra Nasim Khan received MCS and MSCS degree from COMSATS Institute of Information Technology in 2014 and 2016. She joined COMSATS Attock campus, Pakistan as a lecturer in 2017. Her specialization area is machine learning and smartphone security.

Khan Muhammad (S'16) is a Research Associate at Intelligent Media Laboratory, Sejong University, Republic of Korea. He has authored over 30 papers in peer-reviewed international journals and conferences in the areas of image and video processing, information security, image and video steganography, video summarization, diagnostic hysteroscopy, wireless capsule endoscopy, computer vision, deep learning, and video surveillance.

Jong Weon Lee received M.S. degree in Electrical and Computer Engineering from University of Wisconsin at Madison in 1991, and Ph.D. degree from University of Southern California in 2002. He is presently Prefessor of Department of Software at Sejong University. His research interests include augmented reality, human-computer interaction and serious game.

Zhihan Lv is an engineer and researcher of virtual/augmented reality and multimedia major in Mathematics and Computer Science, having plenty of work experience with respect to virtual reality and augmented reality projects, engaging in the application of computer visualization and computer vision. His research application fields widely range from everyday life to traditional research fields (i.e., geography, biology, medicine).

Sung Wook Baik is currently a Full Professor and Dean of Digital Contents at Sejong University. He received his Ph.D. degree in information technology engineering from George Mason University, USA in 1999. He served as professional reviewer for several well-reputed journals. His research interests include computer vision, multimedia, pattern recognition, machine learning, data mining, virtual reality, and computer games.

Peer Azmat Shah is specialized in Computer Networks specially design, implementation, and operation of Future Internet and deeply involved in the research and policies surrounding the Internet. He is Assistant Professor and Head of Department in Computer Science Department at COMSATS IIT, Attock Pakistan since September 2014. He is also leading the Internet, Communications and NETworks (ICNet) research lab at COMSATS.

Khalid Mahmood Awan is specialized in Computer Networks specially resource management in wireless networks, security requirement of networks. He is an Assistant Professor and Deputy Head of Department in Computer Science Department at COMSATS Institute of Information Technology, Attock Pakistan since September 2016. He is also member of Internet, Communications and NETworks (ICNet) research lab at COMSATS.

Irfan Mehmood is an Assistant Professor in Sejong University. He conducts research in a number of basic and applied areas such as image and scene segmentation, motion and video analysis, perceptual grouping, shape analysis and object recognition. His research methods emphasizes the use of segmented (part- based) symbolic descriptions of objects.

Please cite this article as: Z.-U. Rehman et al., Machine learning-assisted signature and heuristic-based detection of malwares in Android devices, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.028