Studies in Indian Place Names
(UGC Care Journal)

ISSN: 2394-3114
Vol-40-Issue-60-March -2020

Malware Detection using Machine Learning for Android Mobile Phones

H Ganesh **Dr S.Sharmila** **Sathish Kumar A P and Guru Prasath S**
Senior Lecturer HoD
Department of Computer Science Department of Electronics and Communication Engineering
NPA Centenary Polytechnic College PSG Polytechnic College, Coimbatore

***Abstract*—In this paper a different method for detecting android malwares is proposed. Here we use machine learning algorithms as opposed to conventional signature based method. To improve the accuracy we use hash functions like MD5, SHA 256, SHA1 along with the supervised machine learning methodology. Three classifiers namely Random Forest, Decision Tree and Logistics Regression are used. Three classifiers are used because we developed a new algorithm with these classifiers which help to reduce the false negative rate. To train these classifiers 19 static features which are highly distinguishable between malware and benign files are used .After the model fitting the classifier the detector is tested with many tests APK files and effectiveness of the detector is measured.**

Keywords— MD5, SHA 256, SHA1, machine learning algorithm, Random Forest, Decision Tree, Logistics Regression, APK file.

I. INDRODUCTION

Today's generation is rightly called digital age. Every field started using electronic computation which is more reliable and faster than humans or any mechanical instruments. Information which must be handled by government or any non government organizations is being digitalized. Today millions of computers are interconnected with each other using networks. Like crime happening in real world crimes also happen in virtual world using these networks in a wrong way. Crimes may include unauthorized access of data in server, bank accounts or even threat to availability of systems by crashing or encrypting them. These are called Cyber crimes and to prevent this cyber security aroused.

The hacker can use different methods to hack a system like social engineering, exploiting, using malware and much more. Using malwares is being more prevalent today. Malwares are software which is intended for malicious use. These attacks are great headaches for organizations. In past many anti-malware companies used signature-based detection and behavioral based detection. Many malwares can trespass these methods. There is realization among companies to use machine learning for malware detection. With the advancement of technology every individuals are having smart phones. With the advent of social media and online money transactions, mobile phones are becoming hackers bunny. Hence it is important to have anti malwares even in mobile phones to prevent from these kinds of attacks.

II LITERATURE REVIEW

Due to vulnerabilities of signature based detection recent research are based upon machine learning techniques. While dynamic approach may be more accurate but it is too costly and time consuming for end user application [1]. Hence static analysis is better for normal mobile phone users.

The classifier for machine learning approach needs lots of data sample's feature vector for training. Lots of researches focuses on very limited set of features (lesser than 3 for average).Naser Peiravian et al [2] uses permissions and frequency of API calls as features. They quantify their features by detecting the presence or absence of each and every permission from the list of 130 and API calls from each and every 1326 leading to total of 1156 features. But this method is vulnerable because of many unwanted features and also today's malware developers can add any extra permissions or API calls without any compromise to performance just to cheat the detector. Pankaj Kohli [6] used only frequency of critical API calls for detection.

ISSN: 2394-3114
Vol-40-Issue-60-March -2020

Westyarian et al [3] used frequency of API packages as features. Unlike Naser Peiravian et al they used APIs class only re[redacted] for permissions. Based on this cr[redacted]ey used frequency of 16 API classes as features. Richard Killam [7] et al use frequency of string literals as their features.

Mahmood Fazlali et al [4] uses all normalized 227 dalvik opcode's frequency as features to detect metamorphic malwares whereas Gerardo Canfora [5] et al uses frequency of 6 classes of opcodes that represents the alteration of control flow as features based on assumption that the complexity of goodware is more than malwares.

Moreover using opcodes as feature is used to find apps with less optimization, which is commonly noticed feature in metamorphic and polymorphic malwares. APK analyzers, IDA pro Disassembler, APK parser, Androguard were the mostly found applications for disassembling the APK files. The dissembled file is used for feature extraction and feature selection.Alejandro Martin et al [8] developed a software tool called Andropytool.This tool integrates multiple tools for dynamic, static and prestatic analysis. It uses Androguard for static and prestatic analysis.Datasets were commonly collected from open public repositories like VirusTotal.com, VirusSign.com etc…for malwares and apks from play store and other reliable sources for goodwares. The commonly used classifiers are random forest, decision tree (J48), support vector machine, naïve bayes and logistic regression. Some even used bagging classifier.

Alejandro Martin et al used ensemble classifier consisting of random forest, decision tree and KNN with same weight. In most of the researches random forest provided better results than others. In some researches it even provided up to 99.5% accuracy. But ensemble classifier and bagging classifier were less superior to conventional ones. Gerardo Canfora [5] recommended the random forest classifier.

III METHODOLOGY

The following figures- figure1 and figure 2 shows the process flow of conventional detection method using machine learning algorithm and our proposed method respectively.

A .Dataset collection

We have taken 100 APK malware samples from github [11] and 100 benign APK files from various sources like apps from Google play store. As we have taken malware and benign files in 1:1 ratio there is no class imbalance problem.

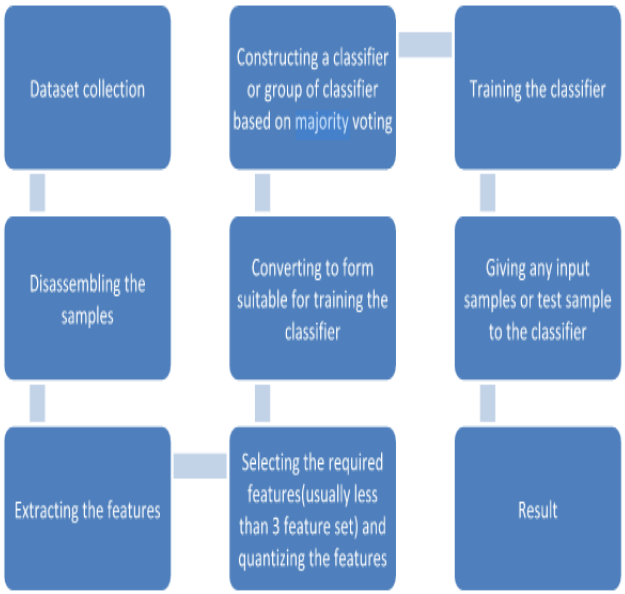
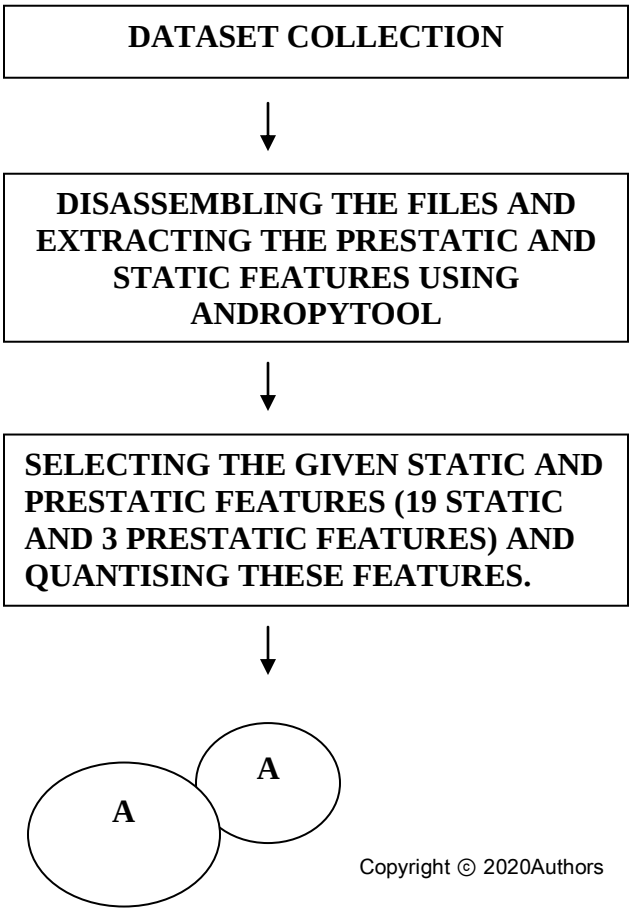
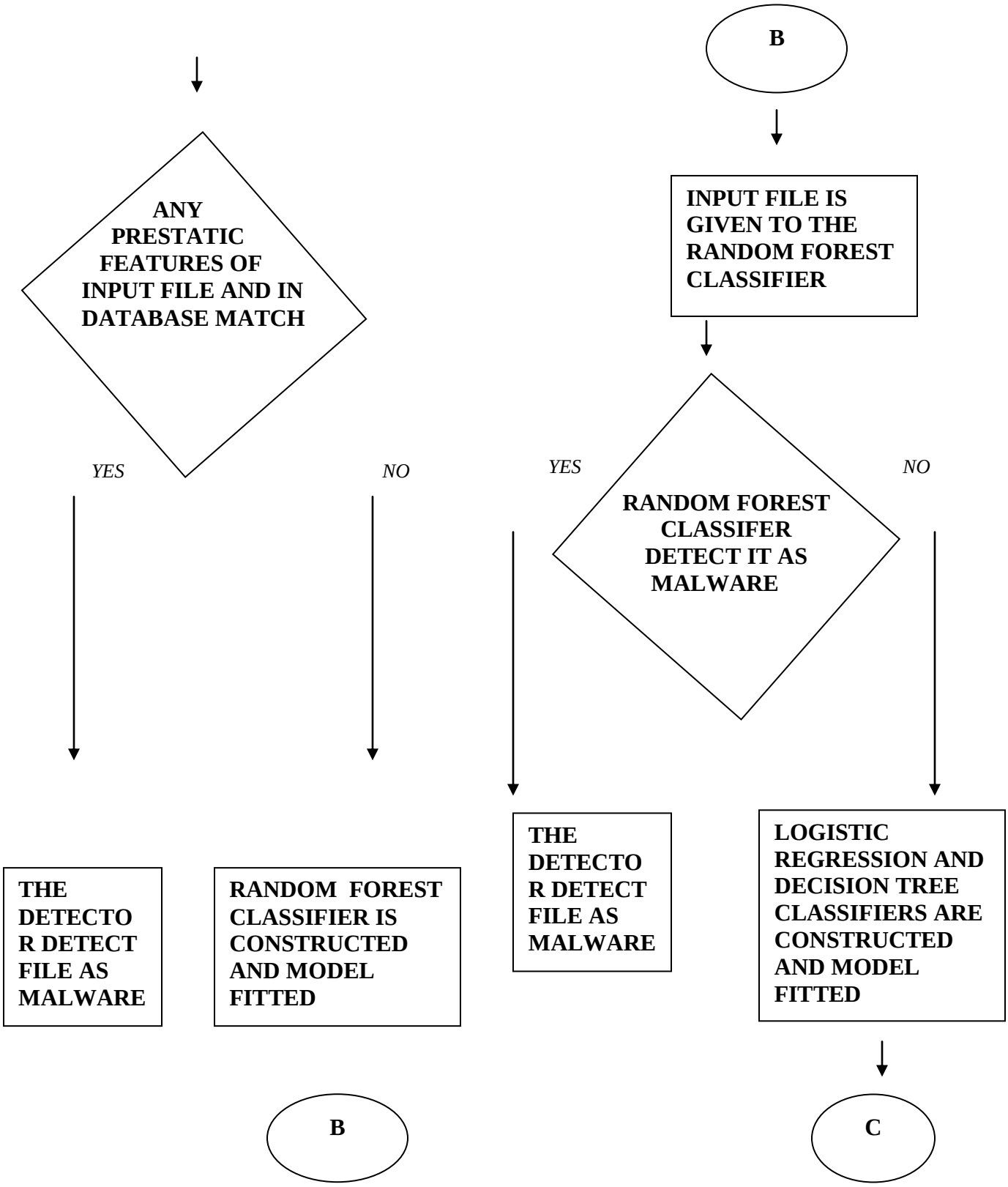


Fig. 1. Process flow of conventional detection method using machine learning



ISSN: 2394-3114
Vol-40-Issue-60-March -2020



ISSN: 2394-3114
Vol-40-Issue-60-March -2020

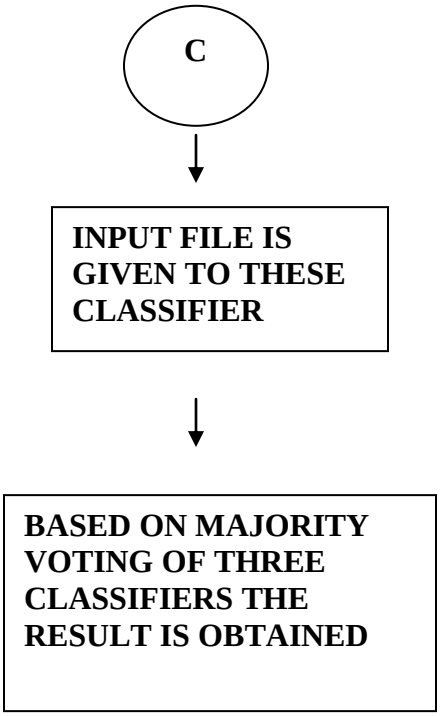


Fig. 2Process flow of proposed method.

B. Feature extraction

AndroPyTool [8] is a tool for the extracting of both, static and dynamic features from Android applications. It aims to provide Android malware analysis with an integrated environment to extract multi-source features able of modeling the behavior of a sample and that can be used to discern its nature, whether malware or good ware. The pre-static analysis is focused on extracting meta-information, features from the compressed apk file which do not require code inspection. The static analysis step is employed to extract more detailed features, such as declared Application Programming Interface (API) calls or permissions required. AndroPyTool integrates different analysis tools and Android applications processing tools, in order to deliver fine-grained reports drawing their individual behavior and characteristics. This framework, which has been implemented as a Python project, takes the apk file to be analyzed as input. Here in this proposed method this tool is used to extract static features and hash functions of the train and test APK files.

C. Feature selection

Proposed malware detector uses multiple features to detect whether the user given APK file is malware or not. This uses 19 static features along with 3 hash functions. These are the following:

Hash function

A hash function is any function that can be used to map data of arbitrary size to fixed size values. The values are used to index a fixed size table called a hash table. Hence hash function can be used to identify a sample. Here we do mere signature kind of analysis by comparing the following hash functions of the data with the samples in dataset which is used to train the classifier. Advantage of this method is if the input APK file is same as the one in database then output can be obtained instantly and also if a malware in the dataset is more unique than other malware in the dataset then its influence on classifier becomes very low. When the same file is given as test file it may produce wrong results. To avoid that cross checking with the help of hash function is used.

MD5

The MD5 message - digest algorithm is a widely used hash function producing a 128-bit hash value.MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as md5sum checksum for the files, so that a user can compare the checksum of the downloaded file to it.

SHA256

The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. A cryptographic hash is like a signature for a text or a data file. SHA-256 algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash.

SHA 1

A 160-bit hash function which resembles the earlier MD5 algorithm.

Permissions

Using permissions as feature is conventional yet finds most of the malware. Usually this feature is quantified by taking in account all the available permissions as features [2] or using set of critical permissions like read_phone_state which allows read phones information like phone number, ongoing calls and network information, send_sms, receive_sms and some more as used by alejadro martin et al [8].but here in our

ISSN: 2394-3114
Vol-40-Issue-60-March -2020

detector number of critical permissions is taken as a single feature. This feature is quantified by counting the number of critical permissions for the test and train apk file. This is based upon the fact that number of critical permissions used by malware is greater than goodwares. Following table I the list of critical permissions [9] on android platform used by our detector:

TABLE I . LIST OF CRITICAL PERMISSIONS

SNO	CRITICAL PERMISSIONS
1	READ_CALENDAR
2	WRITE_CALENDAR
3	CAMERA
4	READ_CONTACTS
5	WRITE_CONTACTS
6	GET_ACCOUNTS
7	ACCESS_FINE_LOCATION
8	ACCESS_COARSE_LOCATION
9	RECORD_AUDIO
10	READ_PHONE_STATE
11	READ_PHONE_NUMBERS
12	CALL_PHONE
13	ANSWER_PHONE_CALLS
14	READ_CALL_LOG
15	WRITE_CALL_LOG
16	ADD_VOICEMAIL
17	USE_SIP
18	PROCESS_OUTGOING_CALLS
19	BODY_SENSORS
20	SEND_SMS
21	RECEIVE_SMS
22	READ_SMS
23	RECEIVE_WAP_PUSH
24	RECEIVE_MMS
25	READ_EXTERNAL_STORAGE
26	WRITE_EXTERNAL_STORAGE

OPCODES

An opcode is referred as operation code which instructs the system to execute the commanded job. The code optimization of goodwares tend to be better than malwares [5]. Also the code complexity of goodwares are tend to be greater than the malwares [5]. Moremover using opcodes of a file even its morphs could be found [4] .Based on these facts opcodes are used as feature for malware detection. Here the method used for quantization of feature is combination of work done by Mahmood Fazlali et al [4] and Gerardo Canfora et al [5].

Here only 6 features that containing the opcodes that transfer the control flow are used. They are given in table II

TABLE II. LIST OF PCODES TAKEN FOR FEATURE

MOVE	IF	INVOKE	JUMP	PACKED SWITCH	SPARSE SWITCH
move-wide	if-eq	invoke-virtual	throw/range	packed-switch	sparse-switch
move-object/from16	if-ne	invoke-super	goto target	-	-
move-object/16	if-ltz	invoke-direct	goto/16target	-	-
move-exception	if-lt	invoke-static	goto/32target	-	-
move/from	if-gt	invoke-interface	-	-	-
move/16	if-ge	invoke-e-virtual/range	-	-	-
move-wide/16	if-nez	invoke-direct-empty	-	-	-
move-object	if-lez	invoke-super-quick	-	-	-
move-object/from16	if-le	invoke-super-quick/range	-	-	-
move-result-wide	if-gtz	invoke-quick/range	-	-	-
move-result-object	if-eqz	invoke-static/range	-	-	-
Move	if-gez	invoke-super/range	-	-	-
move-wide/from 16	check-cast	invoke-interface-range	-	-	-
-	instance-of	invoke-virtual-quick	-	-	-
-	new-instance	invoke-virtual-quick/range	-	-	-

The each feature is quantified by taking in account of percentage of sum of occurrence of each opcode's occurrence with respect to total sum of occurrence of each opcode in the above table.Normalization of opcode=(total occurrence of opcode's occurrence in a APK/ total sum of

ISSN: 2394-3114
Vol-40-Issue-60-March -2020

occurrence of each opcode in the above table in a APK)*100.

API calls

An Application Program Interface (API) is a set of procedures and tools for building software applications. These processes make it easier for developers to use certain technologies in building applications. Using the API calls an APK file interact with underlying operating system and other apps. Unlike some researches which use all the available API calls as features here only the features which are distinguishable is used as it can avoid any unwanted API class without any compromise in performance which can confuse the classifier. According to Westyarian et al [3] total number of API calls called by a application is also a deciding factor. Hence the total number of API calls called by a APK file and six other distinguishable API calls is taken as features. Following are the feature in table III used based on API calls.

TABLE III. LIST OF API CALLS TAKEN AS FEATURE

S.NO	FEATURES BASED ON API CALLS
1	Total number of API calls
2	android.content.Context
3	android.view.View
4	android.widget.TextView
5	android.content.Intent
6	android.content.res.Resources
7	android.os.Handle

API packages

API calls form a useful mechanism to identify relevant deference between samples with good or non-legal intentions. In this study, these calls are grouped by the API packages in which they are defined, in order to depict general characteristic patterns .In general, both types of samples exhibit a similaruse of the most common API packages. For instance, theandroid.app and java. lang packages, which are present inall samples, include the most basic functionality of the Android environment and Java language features respectively. An important difference can be found in the android. Telephony groups of API calls[8]. While 83% of the malicious samples invoke calls contained in this package, this number is reduced to 63% in the benign ware set. Hence android. Telephony is chosen as a feature based on API package.

Intents

Intents are a key communication element in the Android environment. Basically, they allow to ask permission to the system to run another application component. Thus, Intents describe the actions that an application can take. For instance, an Intent could demand actions such asmaking a phone call or taking a picture. There are noticeable patterns which can be inferred from this ranking. Only distinguishable intents are used as feature. Table IV shows the features based on intents.

TABLE IV. LIST OF INTENTS TAKEN AS FEATURE

S. NO	FEATURES BASED ON INTENTS
1	android.intent.action.BOOT_COMPLETED
2	android.intent.action.USER_PRESENT
3	android.provider.Telephony.SMS_RECEIVED
4	Android .intent.action.PACKAGE_ADDED

Classifier

Most of the researches show that the random forest has better accuracy than other classifier. Gerardo Canfor et al [5] recommended random forest classifier many ensemble classifier and lagging classifier fail to complete with the accuracy of random forest drawback of using ensemble classifier with mere majority voting is that the result may be influenced by inferior classifier which is given same weightage of superior classifier like random forest.to avoid this and to reduce the false negative rate a new technique is used. That is the result of random forest classifier is listened first. If it tells the input apk as malware that the apk is tagged as malware that the apk is tagged as malware else if it tells it as good ware then the final decision is taken using the majority voting of all three classifier with same weigthage advantages of this technique is that final decision of being good ware is not influenced by the inferior classifier and false negative rate is reduced.

IV RESULTS

Following are some of the parameters using which the classifier is analyzed. For sake of analysis the known set of dataset is divided

ISSN: 2394-3114
Vol-40-Issue-60-March -2020

into test and train in ratio 4:1. The train samples are model fitted into the classifiers where test sample is used as input to the classifier and the output of the classifier is verified with the known test samples.

A. Accuracy

It is ratio of number of correctly detected files to the number of total number of analysed files by the detector. It is also expressed in terms of percentage. Here in our detector we got maximum accuracy of 97.3%.

B. False positive rate

The false positive rate is the ratio of total number of wrongly detected samples as malware to total detected samples as malware. In our detector we got maximum FPR as 2.45%

C. False negative rate

The false positive rate is the ratio of total number of wrongly detected samples as good ware to total detected samples as good ware. In our detector we got maximum FNR as 1.04%

V SUMMARY

Here for our dataset malwares are collected from open public repository in GitHub[11] and benign apps are taken from APK pure website [12]. The sample consists of 50 malwares and 50 benign apps. As our dataset consists of malware and good ware in ratio 1:1 there is no class imbalance problem. The features are extracted using a software tool called andropytool. It disassembles the given APK and gives pre static and static features in JSON format. The most important features which distinguish malware from good ware are taken from the output of andropytool. The selected features of 100 APK are converted to csv format for the purpose of training the classifier. Here we used three classifiers namely Random Forest, Decision Tree and Logistics Regression. These three classifier is used because our new algorithm with these classifiers helps to reduce the false negative rate. After the model fitting the classifier the detector is test with many tests APK and effectiveness of the detector is measured. Finally the detector can be used to test the unknown user given APK.

VI ACKNOWLEDGEMENT

We would like to thank Dr.S.Sharmila, Head of the Department, ECE, PSG Polytechnic College for enlighting us and guiding us thought the process. We also like to thank all researchers and persons behind the works mentioned in the references.

VII REFERANCES

[1] Mohammed S. Alam et al, “Random Forest Classification for Detecting Android Malware”, IEEE International conference on Green Computing and Communications and IEEE Internet of things, 2013.

[2] Naser Peiravian and Xingquan Zhu, “Machine Learning for Android Malware Detection Using Permission and API Calls”, IEEE 25th International conference on tools with Artificial Intelligence, 2013.

[3] Westyarian, Yuseph Rosmansyah and Budiman Dabarsyah, “Malware Detection on Android Smart phones using API Class and Machine Learning”, The 5th International Conference on Electrical Engineering and Informatics, Bali ,Indonesia,11.10.2015.

[4] Mahmood Fazlali et al, “An efficient algorithm for Detecting Metamorphic Malware by using Opcode Frequency Rate and Decision Tree”, International journal of information security and privacy, July 2016.

[5] Gerardo Canfora Francesco Mercaldo and Aaron Visaggio, “Mobile Malware Detection Using Op-code Frequency Histogram”, Conference paper, May 2015.

[6] Pankaj Kohli et al, “Signature Generation and Detection of Malware Families”, Center for Security, Theory and Algorithmic Research (C-STAR), IIIT Hyderabad.

[7] Richard Killam, Paul Cook and Natalia Stakhanova, “Android Malware Classification through analysis of string literals”, University of New Brunswick.

[8] Alejadro Martin, Raul Lara Cabrera and David Camacho, “Android malware detection through hybrid features fusion and ensemble classifiers: The andropytool framework and omnidroid dataset”, article in Information Fusion, December 2018.

[9] List of normal and critical android permissions, https://stackoverflow.com/ questions/36936914/list-of-android-permissions-normal-permissions-and-dangerous-permissions-on-api

[10] List of Dalvik opcodes used in android pallergabor.uw.hu/androidblog/dalvik_opcodes.html

[11] Malware malware samples repository https://github.com/ashishb/android-malware

[12] Benign android file samples https://m.apkpure.com/