Research Journal of Applied Sciences, Engineering and Technology 12(7): 732-741, 2016
DOI:10.19026/rjaset.12.2749
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 5, 2015 Accepted: November 4, 2015 Published: April 05, 2016

Research Article

Enhanced Permission Based Malware Detection in Mobile Devices Using Optimized Random Forest Classifier with PSO-GA

M. Sujithra and G. Padmavathi
Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, India

Abstract: Smartphones and mobile devices are rapidly growing with their popularity as a part of global infrastructure powered communication system. As mobile devices become ubiquitous, used for a wide variety of application areas like personal communication, data storage, accessing online information, making payment, etc. The tremendous growth of smartphone usage makes it a target for malicious attackers to propagate malware attacks. Increased demand for mobile devices is due to the huge availability of applications that can be downloaded and installed easily on these devices. It is difficult for the general users to differentiate between the set of permissions which are potentially harmful and those which are not. This paper proposes to solve these issues by enhanced machine learning based malware detection framework using optimization algorithms. New classifier is developed by integrating GA and PSO with Random Forest algorithm. The Outcome from this paper is a new MSGP Malware Detection System consisting of MSGP-MDS Classifier. This reveals that classification of Android APK files using PSO plays a critical role in realizing higher accuracy with the minimum computation resource requirement.

Keywords: Android, goodware, machine learning, malware, malware detection, optimization technique

INTRODUCTION

Smartphones are playing important role in everyday lives since they enable everyone to access a large variety of different services from any place. The adoption of smart phone is rapidly increasing which is directly linked to the improved computational power and other utility functions. Smartphone's offer various capabilities of traditional personal computers and in additionally provide a large selection of connectivity options like IEEE 802.11, Bluetooth, GSM, GPRS, UMTS, EDGE, 3G, 4G, HSDPA, HSPA (plus) and LTE. As part of utilizing mobile devices, certain sensitive data such as contact lists, passwords and credit card numbers are stored on these mobile devices. Additionally to a pre-installed mobile operating system like Blackberry OS, Symbian OS, iOS, Android and Windows Mobile, most Smartphone's additionally support Wi-Fi connectivity, (Sujithra and Padmavathi, 2012) in order that users can access the Internet to download and run numerous third-party applications. Although these capabilities provide for useful service to the users, where wide range of devices exchange data with each other thus open up serious security and privacy concerns. The tremendous growth of smartphone usage makes it a target for malicious attackers to propagate malware and perform other malicious attacks.

Malware, as a malicious application that can be installed on mobile devices, with the intention of breaching device security policy with respect to confidentiality, integrity and availability of data. The malware comes in different forms such as a virus, Trojan horse, spyware, adware or trapdoor etc. Malware has proven to be a serious problem for the mobile platform because malicious applications can be distributed to these devices through an application market. Users can download/upload the APK files from third party servers and can install into their mobile devices. Most of the applications from trusted sources are not malware, but the third party server providing malware in modified APK. So before the applications are being installed in the smart phones they can be detected whether they are malware or goodware (ESET Labs, 2013). To mitigate these security threats, various mobile specific Detection Systems have been recently proposed. The presence of a malware in android applications can be detected by using any one of these three techniques. They are:

- Attack or invasion detection
- Misuse detection (signature-based)
- Anomaly detection (behavior-based)

Corresponding Author: M. Sujithra, Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, India

Among these three techniques, the anomaly or behavior based detects the malware with the use of the permissions. Anomaly detection refers to detecting patterns in a given dataset that do not conform to an established normal behavior. The proposed methodology monitors various permissions based features obtained from the android applications and analyze these features by using machine learning classifiers to detect whether the application is goodware or malware. Further the proposed methodology exploits optimization techniques in classification of normal and malware applications with high detection rate. Machine learning is a branch of artificial intelligence that focuses on the development of algorithms that allow devices to reason and decide based on data. Machine learning algorithms can commonly be divided into three different types: supervised learning, unsupervised learning and semi-supervised learning (Fedler *et al*., 2013). Android applications can be properly labeled, so supervised machine learning methods for detection of android malware applications is proposed. Each application must declare what permissions it requires before installed. The mechanism warns the user about permissions an app requested before installed and hopes the user makes the right choice. Extract permission features from the application files and use decision tree supervised machine learning classifiers (RF, CART and J48) to detect malicious applications (Wei *et al*., 2012).

In this framework, the android applications on android market are downloaded and decompressed into the contents of Android applications. The proposed method is based on the characteristic analysis of Android manifest files and is effective for detecting malware. The AndroidManifest.xml and classes.dex files are only selected because these two files contain the necessary permissions features. Android malware applications can be detected by using machine learning approaches. To address the problem, extract android permission features from the application files and use decision tree classifiers (RF, CART, J48) to detect malware in malicious applications. Current techniques in malware classification do not give a good classification result when it deals with the new and unique types of malware. For this reason, the proposed methodology is enhanced with the usage of optimization techniques such as Genetic Algorithm and Particle swam Optimization Algorithm to optimize the malware classification system (Garcia *et al*., 2006). The contribution of the paper includes the enhancement of the optimized Random Forest Classifier. This reveals that classification of Android APK files plays a critical role in realizing a higher detection rate with the minimum computation resource requirement.

LITERATURE REVIEW

Android malware applications have been rapidly rising and there are several approaches to detect these malware applications. Various approaches have been proposed by different authors for detecting malware in android mobile devices based on their permissions. Some of them are discussed below.

Aung and Zaw (2013) monitored various permissions used features and events obtained from the android applications and analyses these features by using machine learning classifiers to classify whether the application is goodware or malware.

Hein (2014) presented the permission based malware protection model for Android application and then uses the self-organizing feature map algorithm. This is express to make small subsequent adjustments of the protection level and to improve the accuracy of the android permissions.

Sanz *et al*. (2013) presented PUMA, a new method for detecting malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself.

Xie *et al*. (2010) proposed a behavior-based malware detection system (pBMDS) that correlates user's inputs with system calls to detect anomalous activities related to SMS/MMS sending.

Abela *et al*. (2013) Gave the capability to classify unknown applications based from its data, can be used

Table 1: Summarizes the significant literatures reviewed for malware detection system

Year	Author	Techniques used	Metrics
2014	Chit La Pyae Myo Hein	Permission based malware protection for Android applications (SOM)	True positive ratio, false positive ratio, Total accuracy
2013	Abela, Kevin Joshua L. Angeles, Don Kristopher E, Delas Alas, Jan Raynier P, Tolentino, Robert Joseph, Gomez, Miguel Alberto N.	Behavior based malware detection (Naïve Bayes algorithm, decision tree algorithms)	True positive Rate, false positive rate, ROC area.
2013	Zarni Aung, Win Zaw	Permission based malware detection.(J48, CART, random forest)	True positive, false positive, true negative, false negative, true positive rate, false positive rate, overall accuracy.
2013	Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, Gonzalo Alvarez.	Permission based malware detection (Machine-learning classifier, k-fold cross validation)	Accuracy, false positive rate, true positive rate.
2010	Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, Sencun Zhu	Permission based malware detection (Hidden Markov Model)	Accuracy, false positive.

to categorize different Android applications in the market and to differentiate whether the application is goodware or malware using behavior based analysis. Detection of malware using different techniques and metrics is listed in Table 1.

PERMISSION BASED MALWARE DETECTION SYSTEM

In this proposed methodology, Machine Learning Classifiers and Optimization techniques are used to analyze and classify the malware applications by comparing the permissions extracted from the applications which are labelled in the dataset. In summary, our main findings are extraction of features from the manifest file of android applications based on the permissions. Selection and Reduction of extracting features are done. Machine learning classifiers are used for the classification and detection of malicious applications. The detection rate of the classifiers is improved by optimization techniques (Rastogi *et al*., 2013).

Feature extraction: Features are the attributes used for defining the permission characteristics of an application. For any downloaded Android application, retrieve the features from the corresponding application package file. Analyze the Manifest file of an application and identify real permissions required by the application. The values of selected features are stored as a feature vector, which is represented as a sequence of bits (0's or 1's). A feature set can be specified as a feature vector, which includes all the permissions that are requested from the user. This framework uses a feature extraction tool written by python script file to extract android permission features (Damopoulos *et al*., 2011). The proposed framework is shown in Fig. 1.

Permissions are requested by an application during the installation process to grant access to various features and functionalities on a device. Currently there are 124 unique permissions which are categorized into 11 top level groups. These permissions are displayed before any application is installed and can also be viewed post installation. The downfall is that users cannot be expected to understand all 124 permissions or the associated risks with a few specific permissions and also it is impossible for users to know which permissions are actually needed by an application.

Every application must have an android Manifest.xml in its root directory. The manifest presents essential information about the application to the Android system. The features in each Android application are extracted through the following steps:

- Download the goodware and malware applications available. Decompress the application (.apk) file by the reengineering process and separate it into its various component files.
- One among the files is the Android Manifest.xml file. This xml file has various permissions contained in it. The permissions of the XML file are extracted and converted into binary form (0 or 1).
- The binary bit of the feature is set valid (1) if the permission is present in the apk file else the bit is set as invalid (0). These permissions form the features through which the dataset is built. Figure 2 is the overall process of automatic feature extraction. The few sample permissions are described in Table 2.

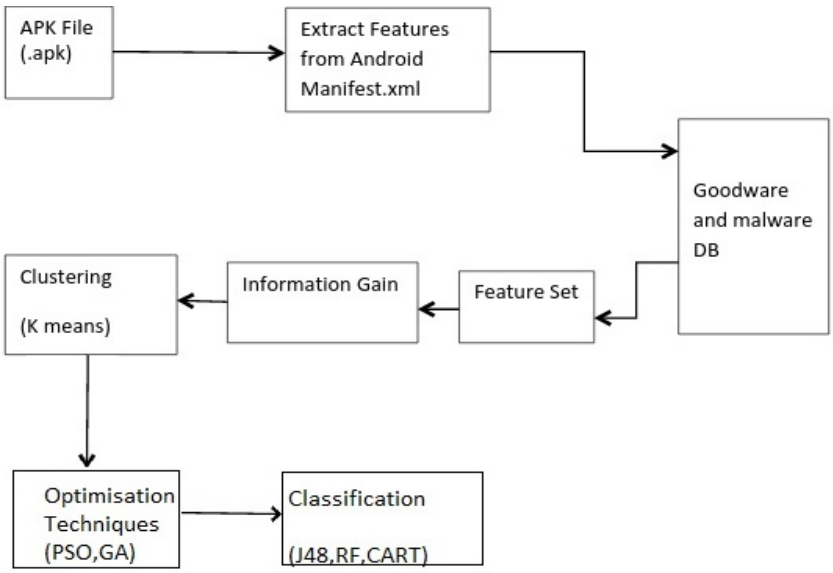

Fig. 1: Permission based malware detection methodology

Table 2: List of permissions on an APK file

Permission	Usage
Android.permission. PROCESS_OUTGOING_CALLS	The application allows the user to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
Android.permission.RECEIVE_SMS	Allows an application to monitor incoming SMS messages, to record or perform processing on them.
Android.permission.SET_PROCESS_LIMIT	Allows an application to set the maximum number of (not needed) application processes that can be running.
Android.permission.CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.

Finally the dataset is formed which is saved in a text format:

0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, goodware

A sample Dataset generated from features of a goodware application.

0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, malware

A sample Dataset generated from features of a malware application.

Feature selection: Feature selection methods are used for reducing the dimension size of a dataset by removing the features (attributes) which are not beneficial to be used in the analysis. Efficient feature selection methods introduce performance gains by reducing the dataset size and the time spent in classification analysis. These adverse effects are even more crucial when applying on mobile devices, since they are often restricted by processing and storage-capabilities, as well as battery power. Information Gain is selected among feature-selection algorithms (Silva *et al*., 2013). It is the method of determining the rank of appropriate feature through the entropy difference between the cases of accurate classification through features. The feature selection is done based on the gain ratio. The features with a higher gain ratio, yield higher optimality to the resultant generation. The features are selected based on the Gain value by referring whether they are greater than 0 and only the features which are greater than 0 is included in the minimized dataset or selected features. According to this Gain value the features are reduced from the original feature set (Bahrololum *et al*., 2009). Entropy should be calculated for each and every feature by the formula given below:

Entropy = $\sum -p_i \log_2 p_i$
Where p_i is the probability of class i.
After the entropy are calculated the gain of a feature is to be calculated as
Gain (S, A) = Entropy (S) - Σ |SV| Entropy (SV) V € Values (A) |S|
[Attribute A on a collection of samples S].

The Feature Selection steps given in Algorithm:

Algorithm for feature selection:

- The entropy and info_split are calculate for each feacher in the dataset.
- The gain ratio is obtained using the entropy and info_split.
- The features with the higher gain ratio are selected and collected into new dataset.

1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, goodware A sample Dataset generated after performing feature selection.

Feature reduction: Number of training samples needed to design a classifier grows with the dimension of the features. A way to reduce the dimension of the features without losing any essential information is needed. The main idea is to define k centroids, one for each cluster. The simple K-means algorithm chooses the centroid randomly from the applications set. The K-means clustering partitions a data set by minimizing a sum of-squares cost function. The selected features are collected in the signature database and divided into training data and test data and used by the standard machine learning techniques to detect android malware applications. K-means clustering uses to group the feature set in clusters. Choosing K-means clustering provides advantages like:

- At least a local minimum of the criterion function is guaranteed and thereby the convergence of cluster on large data sets is accelerated.

- It is a data driven method with relatively few assumptions about the distributions of the underlying data.

Algorithm:

1. Place K points in the space represented by the objects that are being clustered.
2. These points represent initial group centroids.
3. Assign each object to the group that has the closest centroid.
4. When all objects have been assigned, recalculate the positions of the K centroids.
5. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Following provides the pseudocode of clustering:

X: A set of N data vectors
Data set
C_I: Initialized k cluster centroids
Number of clusters,
C: The cluster centroids of k-clustering
random initial centroids
$P = \{p(i) \mid i = 1, \ldots, N\}$ is the cluster label of X
KMEANS(X, C_1) $\rightarrow$ (C, P)
REPEAT
$C_{previous} \leftarrow C_I$;
FOR all $i \in [1, N]$ DO
Generate new optimal paritions
$p(i) \leftarrow \arg\min d(x_i, c_j)$;
$1 \leq j \leq k$
FOR all $j \in [1, k]$ DO
Generate optimal centroids
$c_j \leftarrow$ Average of x_i, whose $p(i) = j$
UNTIL $C = C_{previous}$

MACHINE LEARNING APPROACH

Decision tree classifiers are tree based classifiers for instances represented as feature vectors. They recursively partitions a dataset of records and use a depth first greedy method or breadth first approach. Nodes are used for test features, there is one branch for each value of the feature and leaves specify the category until all the data items belong to a particular class. Decision Trees base the classification of instances by sorting feature vectors. Three machine learning classification algorithms were applied to the data sets: Random Forest (RF), Classification and Regression Trees (CART) and J48 (Kumar and Kumar, 2014).

The random forest algorithm: Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests (RF) are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution of all trees in the forest. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Each tree is independently constructed using a bootstrap sample of data.

The pseudocode of the classifier RF:

- Selected the number of trees to grow and number no larger than number of variables.
- For i = 1 to n tree
- Draw a bootstrap sample from the data call those not in the bootstrap sample the "out-of-bag" data.
- Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree is grow to maximum size and not pruned back
- Use the tree to predict out-of-bag data
- In the end, use the prediction on out-of-bag data to from majority votes.
- Prediction of test data is done by majority votes from prediction from the ensemble of trees.

Forest chooses the classification having the most votes (Glodek and Harang, 2013). A tree is grown by first sampling a random number of N cases in the training set. For each input variable M, a number value m is used for each node to select randomly from the input variable to be used to split a node. After, the generated tree is fully grown as deep as possible.

Classification and Regression Tree (CART): Classification and Regression Trees uses cross-validation or a large independent test sample of data to select the best tree from the sequence of trees considered in the pruning process. The basic CART building algorithm is a greedy algorithm that it chooses the locally best discriminatory feature at each stage in the process. In the implementation of CART, the dataset is split into the two subgroups that are the most different with respect to the outcome. CART partitions the feature space into a set of rectangles and fit a simple model in each one. Then it constructs a binary tree structured classifiers by repeated splits of subsets of the measurement spaces into two descendant subsets. This method assigns a class label for each terminal subset and the resulting partition of x corresponds to the classifier (Denison *et al*., 1998). The pseudocode of the classifier CART.

Let the data be a set of O vector observations, each of length V, such that each observation has one response variable and V-1 predictor variables (supervised learning).

oi = {oil,…, oiV} = {ri, pi1,…, pi(V-1)}

- For all V-1 predictors, order its values (separate into categories) partition the sorted predictor variable at every delta in the sorted values (or by excluding any category) partition the associated response variable in the same way and compute its resulting variance (over two groups)
- Choose the partition which minimizes the response variance over all predictors and thresholds.
- Split the data into 2 pieces on this threshold and repeat steps 1 and 2 on both until some stopping rule is satisfied or each partition contains only 1 data point

J48: The j48 Classification algorithm is inductively learned to construct a model from the pre-classified data set. Each data item is defined by values of the characteristics or features. Classification may be viewed as a mapping from a set of features to a particular class. J48 creates an instance of this class by allocating memory for building and storing a decision tree classifier (Hall *et al*., 2013).

The pseudocode of the classifier J48:

1. Create a root node N
2. If T belongs to the same category C, then return N as a leaf node and mark it as class C
3. If attribute list is empty or the reminder sample of T is less than a given value, than return N as a leaf node and mark it as the category which appears most frequently in attribute list, for each attribution, calculate its information gain ratio
4. Suppose test attribute is the testing attribute of N, then test attribute-the attribute which has the highest information gain ratio in attribution list;
5. If testing attribute is continuous, then find its division threshold
6. For each new leaf node grow by node N

   {
   (a) Suppose T is the sample subset corresponding to the leaf node.
   (b) If T has only a decision category, then mark the leaf node as this category,
   (c) Else continue to implement J45_Tree (T', T'_Attribute list)
   }

7. Calculate the classification error rate of each node and then prune the tree.

Basic Steps in the Algorithm:

- In case the instances belong to the same class the tree represents a leaf so the leaf is returned by labelling with the same class.
- The potential information is calculated for every attribute, given by a test on the attribute. Then the gain in information is calculated that would result from a test on the attribute.
- Then the best attribute is found on the basis of the present selection criterion and that attribute selected for branching.

ENHANCED CLASSIFICATION USING OPTIMIZATION ALGORITHM

Current techniques in malware classification do not give a good classification result when it deals with the new and unique types of malware. For this reason, the usage of optimization techniques, namely Genetic Algorithm and Particle Optimization Algorithm is used to optimize the malware classification system. This new malware classification system also has an ability to train and learn by itself, so that it can predict the current and upcoming trend of malware attack. One of the main goals is to detect and classify the unique malware that has a relationship during the execution. The other goal is to find unique malware that performs the same behavior, but providing different syntax representation. A framework is proposed by combining GA and PSO with the implemented machine learning classifiers.

Proposed methodology-1 genetic algorithm with RF classifier: GA is belongs to the larger class of Evolutionary Algorithm (EA). GA includes the survival of the fittest idea into a search algorithm which provides a method of searching, which does not need to explore every possible solution in the feasible region to obtain a good result. GA also commonly used for a learning approach to solve computational research problem. By tradition, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. In each generation, the fitness of every individual in the population is evaluated. The fitness is usually the value of the objective function in the optimization problem being solved. The fittest individuals are stochastically selected from the current population and each individual's genome is modified to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations have been produced, or a satisfactory fitness level has been reached for the population (Yusoff and Jantan, 2011). A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain is as following:

Evaluate each individuals fitness
Determine population's average fitness
Repeat
    Select best ranking individuals to reproduce
    Mate pairs at random
    Apply crossover operator
    Apply mutation operator

Evaluate each individual's fitness
Determine population's average fitness
Select ntree, the number of trees to grow and mtry, a number no larger than a number of variables
For i = 1 to n tree:
Draw a bootstrap sample from the data.
Call those in the bootstrap sample the "out-of-bag" data.
Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree has grown to maximum size and not pruned back. Use the tree to to predict out-of-bag data.
In the end, use the predictions on out of bag data to form majority votes. Prediction of test data is done by majority votes from predictions from the ensemble of trees.

Proposed methodology-2 particle swarm optimization with RF classifier: Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (Senthilkumar and Kannan, 2014). PSO optimizes a problem by having a population of candidate solutions, here dubbed particles and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. PSO achieve its optimal solution by starting with a group of random solution and then searching repeatedly (Ahandani and Baghmisheh, 2013). This is expected to move the swarm toward the best solutions.

For each particle
Initialize particle
Do
For each particle:
Calculate the fitness value
If the fitness value is better than the best fitness value (pBest) in history
Set current value as the new pBest
End
For each particle:
Find in the particle neighborhood, the particle with the best fitness
Calculate particle velocity according to the velocity equation
Apply the velocity Constriction
Update particle position according to the position equation
Apply the position constriction
Select ntree, the number of trees to grow and mtry, a number no larger than a number of variables.
For i = 1 to ntree:
Draw a bootstrap sample from the data. Call those in the bootstrap sample the "out-of-bag" data.
Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree has grown to maximum size and not pruned back.
Use the tree to predict out-of-bag data.
In the end, use the predictions on out of bag data to form majority votes.
Prediction of test data is done by majority votes from predictions from the ensemble of trees.

EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the proposed framework, collected 1000 including normal applications from android market and malicious applications from the internet site. The dataset used during the evaluation were composed of Android apps collected in this system. These apps were already classified into benign and malicious samples. Out of over 136,000 available apps from Google's official Play Store and out of over 40,000 malicious samples identified by Virus Total, representing 192 malware families, randomly selected 200 distinct apps. In detail, selected 150 benign apps and 50 malicious apps.

In order to perform the evaluation of the proposed mechanism and comparison between the various detection algorithms and feature selection schemes here employed the following standard metrics: the True Positive Rate (TPR) measure, which is the proportion of positive instances classified correctly; False Positive Rate (FPR), which is the proportion of negative instances misclassified; and the Total Accuracy, which measures the proportion of absolutely correctly classified instances, either positive or negative. The performance of the proposed swarm optimized technique over the machine learning techniques comparatively considered were evaluated in terms of the below parameters such as Detection time, True positive rate, False positive rate and Detection accuracy (Ham and Choi, 2013):

True Positive Rate (TPR): Percentage of correctly identified goodware applications:

$$TPR = (TP/TP+FN)$$

False Positive Rate (FPR): Percentage of wrongly identified malware applications:

$$FPR = (FP/TN+FP)$$

Precision value: It is the number of correctly classified positive examples with respect to the number of examples that exist in the system as positive.

$$\text{Precision value} = (TP/TP+FP)$$

Table 3: Experimental results classifiers

Algorithm	TP rate	FP rate	Precision	Recall	Correctly identified instances (%)	Incorrectly identified instances (%)
J48	0.83	0.17	0.87	0.79	83.3	16.7
CART	0.79	0.21	0.86	0.69	79	21
Random forest	0.87	0.13	0.91	0.81	86.8	13.2

Table 4: Experimental results optimized classifiers

Algorithm	Correctly identified instances (%)	Incorrectly identified instances (%)
Random forest	86.8%	13.2%
Genetic algorithm with RF	87%	13%
Particle swarm optimization with RF	88.4%	12.6%

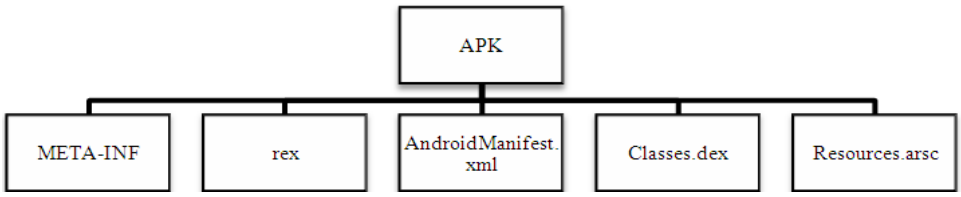

Fig. 2: An example of decompile APK file

Recall: Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved:

Recall = (TN/TN+FN)

Overall accuracy (ACC): Percentage of correctly identified applications

ACC= (TP+TN/TP+TN+FP+FN)

True Positive (TP) is the number of correctly identified goodware applications, False Positive (FN) is the number of wrongly identified goodware applications, True Negative (TN) is the number of correctly identified malware applications and False Positive (FN) is the number of wrongly identified goodware applications.

Table 3 provides the comparison of parameters between J48, CART and Random Forest. The given parameters are True positive rate, false positive rate, Precision Value in (%) and Recall Value in (%) and Accuracy in (%).

Table 4 provides the comparison of Random Forest, Genetic Algorithm and particle swarm optimization using the parameters such as correctly identified instances (Accuracy) in % and incorrectly identified instances in %.

Figure 3 gives the comparison, that random forest has high correctly identified instances of about 86.8% than compared to J48 whose correctly identified instance is 83.3% and CART of correctly identified instance 79%. Again to increase this accuracy, optimization techniques are used.

Figure 4 gives the comparison, that particle swarm optimization has high correctly identified instances of about 88.4% than compared to genetic algorithm of correctly identified instance is 87% and random forest of correctly identified instance 86.8%.

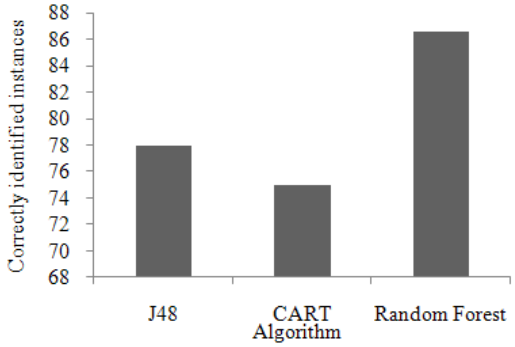

Fig. 3: Experimental results-classifiers

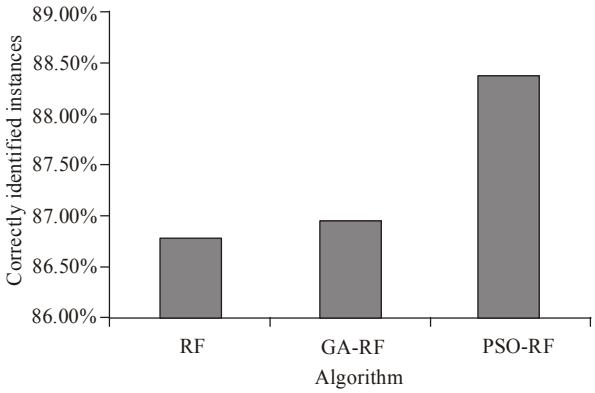

Fig. 4: Experimental results-optimized classifiers

CONCLUSION

A framework for detection of android malware applications using machine-learning techniques has been proposed by extracting permission features from several downloaded applications from android markets. The results were further optimized by optimization techniques to detect the android applications whether it is goodware or a malware application. This paper

proposed the usage of optimization algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) as an approach to optimize Random forest Decision Tree in malware classification. New classifier is developed by combining GA and PSO with RF_DT named as MSGP Malware System (MSGP-MS) Classifier. Using real-world malware and benign applications, experiments were conducted on Android mobile devices. Experimental results obtained from MSGP-MS Classifier with RF are compared and visualized in tables and graphs. MSGP-MS Classifier shows an accuracy increase from RF Classifier. The outcome of this paper is a new MSGP Malware Classification System consisting of MSGP-MS Classifier. This reveals that classification of Android APK files using PSO plays a critical role in realizing higher accuracy with minimum computation resource requirement.

Conflicts of interest: The idea of detecting the malware by machine learning classifiers exits in literature, However the issues are not handled properly. Our proposed methodology handles the pitfalls with the improvisation of results.

REFERENCES

Abela, L.K.J., E.D.K. Angeles, P.D.A.J. Raynier, R.J. Tolentino and N.A.G. Miguel, 2013. An automated malware detection system for android using behavior-based analysis AMDA. Int. J. Cyber-Secur. Digit. Foren., 2(2): 1-11.

Ahandani, M.A. and M.T.V. Baghmisheh, 2013. Hybridizing genetic algorithms and particle swarm optimization transplanted into a hyper-heuristic system for solving university course timetabling problem. WSEAS T. Comput., 12(3): 128-143.

Aung, Z. and W. Zaw, 2013. Permission-based android malware detection. Int. J. Sci. Technol. Res., 2(3): 228-234.

Bahrololum, M., E. Salahi and M. Khaleghi, 2009. Machine learning techniques for feature reduction in intrusion detection systems: A comparison. Proceeding of the 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT'09). Seoul, pp: 1091-1095.

Damopoulos, D., S.A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke and S. Gritzalis, 2011. Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers. Secur. Commun. Netw., 5(1): 3-14.

Denison, D.G.T., B.K. Mallick and A.F.M. Smith, 1998. A Bayesian CART algorithm. Biometrika, 85(2): 363-377.

ESET Labs, 2013. Trends for 2013: Astounding Growth of Mobile Malware. Retrieved from: http://go.eset.com/us/resources/white-papers/Trends_ for_2013_ preview.pdf.

Fedler, R., J. Schutte and M. Kulicke, 2013. On the effectiveness of malware protection on android: An evaluation of android antivirus apps. Technical Report Antivirus Test, Fraunhofer Research Institution for Applied and Integrated Security, Fraunhofer AISEC, pp: 1-35.

Garcia, J.S.D., S.L. Ávila and W.P. Carpes Junior, 2006. Introduction to optimization methods: A brief survey of methods. IEEE Multidiscipl. Eng. Educ. Mag., 1(2): 1-5.

Glodek, W. and R. Harang, 2013. Rapid permissions-based detection and analysis of mobile malware using random decision forests. Proceeding of the IEEE Military Communications Conference (MILCOM, 2013). San Diego, CA, pp: 980-985.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, 2013. The WEKA data mining software: An update. ACM SIGKDD Explor. Newslet., 11(1): 10-18.

Ham, H.S. and M.J. Choi, 2013. Analysis of Android malware detection performance using machine learning classifiers. Proceeding of the International Conference on ICT Convergence (ICTC, 2013). Jeju, pp: 490-495.

Hein, C.L.P.M., 2014. Permission based malware protection model for android application. Proceeding of the International Conference on Advances in Engineering and Technology (ICAET'2014). Singapore, pp: 222-226.

Kumar, A. and S. Kumar, 2014. Decision tree based learning approach for identification of operating system processes. WSEAS T. Comput., 13: 277-288.

Rastogi, V., Y. Chen and X. Jiang, 2013. Droidchameleon: Evaluating android anti-malware against transformation attacks. Proceeding of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13), pp: 1-6.

Sanz, B., I. Santos, C. Laorden, X. Ugarte-Pedrero, P.G. Bringas and G. Álvarez, 2013. PUMA: Permission usage to detect malware in android. Proceeding of the International Joint Conference on CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions. Springer, Berlin, Heidelberg, 189: 289-298.

Senthilkumar, B. and T. Kannan, 2014. Multi-objective optimization of bead geometry and dilution in FCAW process using PSO. Int. J. Appl. Eng. Res., 9(24): 25817-25832.

Silva, L.O.L.A., M.L. Koga, C.E. Cugnasca and A.H.R. Costa, 2013. Comparative assessment of feature selection and classification techniques for visual inspection of pot plant seedling. Comput. Electron. Agr., 97: 47-55.

Sujithra, M. and G. Padmavathi, 2012. Mobile device security: A survey on mobile device threats, vulnerabilities and their defensive mechanism. Int. J. Comput. Appl., 56(14): 24-29.

Wei, X., L. Gomez, I. Neamtiu and M. Faloutsos, 2012. Permission evolution in the android ecosystem. Proceeding of the 28th Annual Computer Security Applications Conference (ACSAC'12). NY, USA, pp: 31-40.

Xie, L., X. Zhang, J.P. Seifert and S. Zhu, 2010. pBMDS: A behavior-based malware detection system for cellphone devices. Proceeding of the 3rd ACM Conference on Wireless Network Security, pp: 37-48.

Yusoff, M.N. and A. Jantan, 2011. A framework for optimizing malware classification by using genetic algorithm. In: Zain, J.M. *et al*. (Eds.), ICSECS, 2011. Part II, Communications in Computer and Information Science, Springer-Verlag, Berlin, Heidelberg, 180: 58-72.