Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 10 (2015) pp. 152-163
© MIR Labs, www.mirlabs.net/jias/index.html

Static Code Analysis of Permission-based Features for Android Malware Classification Using Apriori Algorithm with Particle Swarm Optimization

Olawale Surajudeen Adebayo[1], Normaziah Abdul Aziz[2]

[1]Computer Science Department, International Islamic University Malaysia, and
CSS Department, Federal University of Technology Minna, Nigeria
adebayo.olawale@live.iium.edu.my, waleadebayo@futminna.edu.ng

[2]Computer Science Department,
International Islamic University Malaysia
naa@iium.edu.my

Abstract: **Several machine learning techniques based on supervised learning have been applied to classify malware. However, supervised learning technique has limitations for malware classification task. This paper presents a classification approach on android malware using candidate detectors generated from an unsupervised association rule of Apriori Algorithm. The algorithm is improved with Particle Swarm Optimization that trains three different supervised classifiers. In this method, permission-based features were extracted from Android applications byte-code through static code analysis, selected and were used to train supervised classifiers. Using a number of candidate detectors from an improved Apriori Algorithm with Particle Swarm Optimization, the true positive rate of detecting malicious code is maximized, while the false positive rate of wrongful detection is minimized. The results of the experiments show that the proposed combined technique has better results as compared to using only supervised or unsupervised learners.**

Keywords: Android Malware; Apriori Algorithm; Particle Swarm Optimization; Malware Detection; Static Analysis; Supervised Learning; Unsupervised Learning

I. Introduction

Static code analysis of malware is an analysis of malware code without actually running the code [36]. In this method, researcher acquires malicious code from available sources, decompile the code using a combination of static analysis tools and interpret the equivalent malicious features. The available features in android application include Permission-based features, and API-based features (methods, classes, calls, functions, Activities, Services etc.) [45]. Permission-based features are usually requested from the users by the application before apps can be installed on the android phone. Only after permission is granted then the apps will be installed on the phone. The permission-based features could be categorized into dangerous, normal, and signature. The dangerous permission, however varies in different applications, it could be low-level to high-level dangerous permission. For example, an apps permission that request access to send, modify or delete memory contents of a phone is of high security risk (dangerous) while the one with permission request for network access or prevent phone from sleeping is of less security risk.

Data mining method of detecting malware has been very effective in the classification of malware. This field of study can be classified into supervised and unsupervised learning strategies and several techniques [1]. The strategy or technique to be adopted by an expert for the classification task depends on the nature of data and the problem to be solved, that is whether the output of the data is categorical or numerical. Learning techniques for supervised data mining includes Rain Forest Neural network, decision tree, Bayesian, Naïve Bayes, Classification-based Multiple Association Rule (CMAR) [24].
An unsupervised learning technique is based on clustering algorithm such as k-Nearest Neighourhood and some other clustering algorithms. Supervised learning can be basically used for three purposes namely classification, prediction, and estimation depending on the output of data or whether to determine present or future circumstances.

In this research, association rules mining of Apriori Algorithm is improved and used for automatic candidate generation and selection of android applications' features for effective classification. The original Apriori Algorithm was proposed by Agrawal R. et al [2] in order to address the problem of mining association rules. The need to improve the efficiency of mining of frequent item sets (highest occurring items), by reducing the times of scanning the database and reducing the number of candidate item sets, prompted [3] to propose an improved Apriori Algorithm based on the classic Apriori Algorithm. The basic idea of Apriori Algorithm is to generate the frequent itemsets using iterative method in order to generated rules that meet the minimum confidence to form rule sets and outputs [3].

Android malware is able to compromise the security of information on the smartphone. It is a threat since most facilities available on the conventional operating systems on computer are also present on the android operating system. This has made the security of android phone an important task in order to secure vital information of the users. This security threat on the android smartphone is compounded as a result of several attack vectors and surfaces. Attack vectors are methods through which an attacker carried out its act i.e. electronic mail attachment, clickable URL, and API functions while attack surfaces are target's open flanks or characteristics of a target that makes it vulnerable to attack i.e. web related technology (http, html, css, etc.), piece of code from attacker [37].

The basic ideas in this paper are 1) improving Apriori Algorithm using Particle Swarm Optimization as the selection approach for the classification of android malware permission-based features, and 2) classifying android malware features using an improved Apriori Algorithm as selection technique to show its effectiveness over the original Apriori Algorithm and some other selection techniques for malware classification. Apriori Algorithm task is basically divided into three namely: candidate generation, candidate counting, and candidate selection.

This research adopts Particle Swarm Optimization to improve the generation of candidate detectors (flagbearers) which shall otherwise improve the classification process by maximizing the true positive detection and minimizing the false positive detection. Particle Swarm Optimization is used initially to generate candidates for later stage while Apriori Algorithm is applied for candidate counting and selection in order to have the best set of candidate detectors for the supervised training. The researchers obtained several android applications both good and malicious for the purpose of classification and prediction. The features were extracted from both samples after a thorough analysis of .apk files. Three feature selection approaches were used to select high ranked features from the set of generated features.

The rest of this paper is organized as follows: related works to this research is discussed in section II. Section III discussed the proposed model with its constituent frameworks. In section IV, empirical study, results and conclusion is given to the work. Section V is used to explain the experimental study and discussion. Section VI concludes the discussion of this paper.

II. Related work

A malware is a computer program that has various kinds of malicious intents [4]. Mobile malwares are those malwares designated to operate on the mobile facilities through mobile applications for malicious activities. Android operating system being a flexible and open source operating system on the smartphone has been a major target by malware over time. Malware detector is a model or algorithm developed to detect and contain the dastard effects of malicious program [5]. Machine learning techniques have been widely applied in the classification of malware. The work in [30] used three different features namely: program header, string features, byte sequence features and four classifiers (Naïve Bayes, Rule based classifier, signature based, and Multi-Naïve Bayes classifier) in classifying malware with all other three classifiers outperform signature based method. Another work in [31] combined N-Gram feature with k-nearest neighour classifier for the classification. Researches in [32], [33] have also trained different classifiers using malware features collection and obtained improved performance for different classifier.

API-based android malware detection has been used in [45] where the performance of API features malware detection was compared with permission based detection using four different classifiers. The behavioural malware detection on mobile handset in order to curb the casualty in the mobile community is another detection technique by [6]. Their approach is unique in the definition of application behaviour. Their approach observes the programs' run-time behaviour at a higher level (i.e., system events or resource-access) than system calls of [47] and machine instructions of [48]. This higher-level abstraction improves resilience to polymorphism and facilitates detection of malware variants, as it abstracts away more low-level implementation details. Also, the approach employs a runtime analysis, effectively bypassing the need to deal with code/data obfuscation [49].

Among the recent and leading literatures on the detection of malware on mobile platform include Framework For Analyzing Android Applications (ANANAS) [50] and lightweight Malware Detection System for Android-based mobile devices (ANDROMALY) [51]. ANANAS focused on automated static and dynamic malware analysis using core framework and analysis plugins while Andromaly monitors both the smartphone and user's behaviours by observing several parameters, spanning from sensors activities to CPU (central processing unit) usage and using several features to describe behaviours.

Crowdroid [52] is another android malware detector that uses system calls to detect malicious patterns on the Android phones. It helps users by sending non-personal, but behaviour-related data of each application they use to the central server for malware analysis. A Multi-level Anomaly detector for Android Malware (MADAM) [53] by Gianluca Dini et al. detects malware using machine learning classification and anomaly-based system by concurrently monitor android at kernel-level (machine low level) and user layers (application layer). It combines system calls with the activity monitors and SMS monitors in order to detect malware. An automated behavioural analysis system (AMDA) [54] determines malicious behaviour from benign behaviour through the use of machine learning techniques.

T. Bläsing et al. [55] also develop an Android Application Sandbox system for suspicious software detection using dynamic, single API, clustering and fake API injection techniques. This application only works on an android platform. Suhas Holla and Mahima M Katti [56] discussed Android mobile platform for the mobile application development, layered approach and the details of its security information. Andrew Walenstein et al. [57] proposes an approach for selecting features of mobile malware by using knowledge of malicious program structure to heuristically identify malicious portions of applications.

One of the basic techniques of classifying malware into malicious or benign is data mining. The initial problem of mining association rules was addressed by Agrawal R. et al. [2]. Apriori Algorithm where the generated frequent itemsets were used to generate rules that meet the minimum confidence to form rule sets and output. The research in [5] used an association rules mining of Apriori Algorithm to automatically generate frequent itemsets of program signatures (malware and benign) and extract features from the parsed files for subsequent supervised learning. In another work, Shabtai A. et al. [27] classified games and tools using features extracted from android .apk files of both application.

Due to the challenges of Apriori Algorithm in generating large quatities of itemsets and time consuming in testing and verifying candidate frequent k-items [3], which have resulted to its inefficiency, different versions of Apriori Algorithm have been developed that manifested an improvement in the original algorithm like an improved Apriori Algorithm that addressed the inefficiency in Apriori Algorithm [3]. This research, in an effort to improve Apriori Algorithm for the detection of malicious programs, adopts Particle Swarm Optimization in the candidate generation of detector so as to increase the detection process and reduce false alarm rate.

III. The Proposed Improved Model and Its Associated Frameworks

This proposed improved system is composed of Apriori Algorithm and Particle Swarm Optimization combined in a strategic way with negative border as the fitness function for selection process and signature extraction. The essence of mining association in malware detection system is to generate the best set of features called candidate detectors through unsupervised learning for the supervised training. Association rule could also be used to extract important information from the collected features and to discover useful association rules in the signature. This task can be decomposed into two viz [24]: first, discovering the large itemsets, that is the sets of items that have support s above a predetermine threshold; second, use the large itemsets to generate the signature rules for the features that have confidence above a predetermine threshold.

The Apriori Algorithm consists of three basic steps namely; generate phase, count phase, and select phase. The generate phase generates candidate itemsets repeatedly to discover large itemsets (Large-k-itemsets) using $L_k * L_k$ that meet up with minimup support and confidence [24]. The operation is given as in equation 1. $L_k * L_k = \{A \cup B$ where $A, B \in L_k,$ and $|A \cap B| = k - 1\}$…. (1), where k = 1 then C_k of k-itemsets were generated using equation 2 as candidate in the next iteration. $|L_k *(|L_k| - 1)/k$ ……………………. (2)

Note: $|L_k|$ denotes absolute value of L_k; C_k is the subset of k-itemsets.

The second phase of the algorithm scans the (k-1)-itemsets to count the support the support for every candidate and select a large k-itemsets L_k for which support $s \geq$ min threshold. In the select phase, only candidates whose support meets the mininmum threshold are selected for next phase of candidate generation using minima support and minima confidence. The detector generated by [5] proved not to be effective due to the slow generation of candidate detectors by Apriori Algorithm. Other researches which include [11], [9], [12] have attempted to provide solution to the association problem of detecting malware using Apriori Algorithm of association rule mining.

Particle swam optimization (PSO) was developed by Eberhart and Kennedy [34] in 1995 to address the problem of optimization. The problem was model against the behavior of a group of birds searching for food and follows a particular bird that is nearest to the food. Particle Swarm Optimization has been applied successfully for the generation of candidate detector in negative selection algorithm for spam detection [7], [14], virus detection [8], feature selection [13], [15], [16], anomaly detection [10], [20], intrusion and misuse detection [17] [18], [19]. PSO has also proved to be a successful optimizer in fuzzy system [38], [39], multi-objective problems [40], and tracking system [41]. It was used with rough set by [32] in order to improve the effectiveness and efficiency of selection method. PSO has also evolved in various forms [44], [42], [43] in a bid to improve the original PSO by Kennedy [34].

A. Support of the Rule and Confidence

A mathematical formality of support of the rule and confidence of the rule of association rule used in this research is as follows:

D represents dataset;
P and Q denote itemsets;
Si denotes sequences;
P => Q denotes if a sequence s contain in itemset P then it is also likely to contain in itemset Q.

The threshold minsupp & minconf are parameters specified by user to indicate rule interested in.

Given a minsupp, an itemset P is said to be frequent in the dataset D if support D (p) ≥ minsupp

A sequence whose support satisfies minsupp is called a frequent sequence

$$Support\ of\ the\ rule\ P => Q\ in\ data\ sample\ D\ of\ sequences\ S\ is\ the\ percentage\ (\%) of\ sequence\ S\ in\ D\ that\ contain\ P \rightarrow Q$$

$$Support^D(P \rightarrow Q) => \frac{|\{S' \in D|\ |P \rightarrow Q \subset S'\}|}{|D|}$$

$$Confidence\ of\ the\ rule\ P => Q\ in\ dataset\ D\ of\ sequences\ S\ is\ the\ percentage\ (\%) of\ sequence\ S\ in\ D\ that\ contain\ P\ that\ also\ contain\ P \rightarrow Q$$

$$Confidence^D(P \rightarrow Q) => \frac{Support^D(P \rightarrow Q)}{Support^D(s)}$$

$$= \frac{|\{S' \in D|\ |P \rightarrow Q \subset S'\}|}{|\{S' \in D|\ |P \subset S'\}|}$$

The Original Apriori Algorithm Pseudocode

```
// Input: Database D, minimum support threshold; min-sup
// Output: Frequent itemsets F in D
Let $L_1$ be large 1-itemsets
Let k be counter, the number of instances in the database D
For k = 2
  $L_{k-1} \neq \phi$
  k = k+1
$C_k$ = Itemset.gen {$F_{k-1}$, minsup}; // Randomly generate candidate itemsets

Public Sub Itemset.gen:
For each itemset $I_1 \in L_{k-1}$
    For each itemset $I_1 \in L_{k-1}$
      If ($I_1[1] = I_2[1]$) and ($I_1[2] = I_2[2]$) and … and ($I_1[k-2] = I_2[k-2]$) and ($I_1[k-1] = I_2[k-1]$) Then
            Concatenate $I_1$ and $I_2$ to form p;  // the generated candidates
          If there exist infrequent.subset (p, $L_{k-1}$) Then
               Delete p; // remove infrequent itemsets
          Else insert p into P
          End
       End
    End for
Return C
End Sub

Public Sub infrequent.subset:
 For each itemsets(k-1).subset g of p
   If $g \in L_{k-1}$ then return TRUE else Return FALSE
End Sub

For all transactions $t \in D$
  $P_s$ is a subset of ($P_k$, s)
For all candidates $p \in P_s$
Pc: p = p + 1
End for
$F_k = \{p \in P_s \mid Pc \geq minsup\}$
End for
Return L = set of $L_k$
```

Figure 1. Original Apriori algorithm Pseudocode [2]

B. Optimization of Apriori Algorithm Candidate generator with Particle Swarm Optimization (AA-PSO)

The most important task in Apriori Algorithm is the candidate generation of large k-itemsets with highest frequency and the association of rules. The problem is to generate large k-itemsets that meet the minima support and confidence in a short period of time with efficiency. This paper presents a technique to optimize the generation of large k-itemsets using PSO in order to increase the effectiveness of feature selection, classification and detection model. The particle's velocity and position in an updated standard PSO was given in equations (3) and (4) respectively below:

$$V_{id}(t+1) = wV_{id}(t) + c_1r_1(P_{id}best(t) - x_{id}(t) + c_2r_2(P_{gd}best(t) - x_{id}(t) \quad (3)$$

$$x_{id}(t+1) = x_{id}(t) + V_{id}(t+1) \quad (4)$$

where i = 1, 2, …, n, n represent the number of particles in the swarm, d = 1, 2, …, D, D is the dimension of solution space. $w \in [0,1]$ is the inertia weight associated to the given particle velocity and position to ensure balance between the local and global search best. Also c_1 and c_2 represent the nonnegative learning factor while r_1 and r_2 uniformly distributed random numbers in the interval [0, 1]. The velocity $V_{id} \in [-V_m, V_m]$, where V_m is a maximum velocity predefine by the users in relation to the objective function. In this paper, we used infrequent items otherwise known as negative border or atypical factor as the fitness function in order to reduce the time and space complexity.

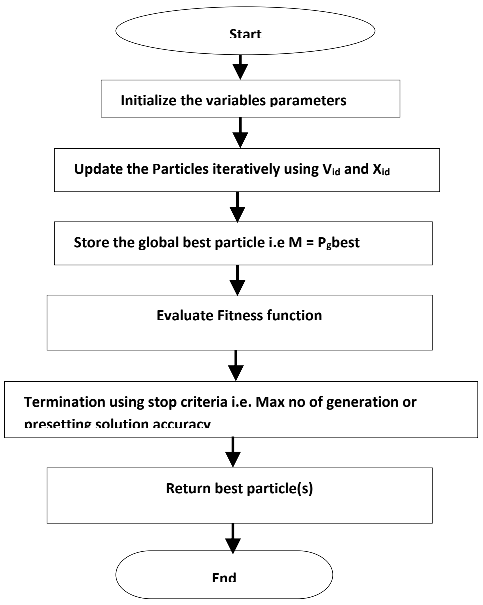

Start
Initialize the variables parameters
Update the Particles iteratively using V_{id} and X_{id}
Store the global best particle i.e M = $P_g$best
Evaluate Fitness function
Termination using stop criteria i.e. Max no of generation or presetting solution accuracy
Return best particle(s)
End

Figure 2. Particle Swarm Optimization model [34]

Table 1. Samples of Common Dangerous Permissions in the Android Applications.

Permissions	Descriptions	Malicious Effects
android.permission.WRITE_CONTACTS	Allows Application to modify the contact data store on the phone	Malicious Apps can use this permission to erase or modify contact data
android.Permission.READ_CONTACTS	It reads contact data	Malicous Apps can use this permission to send data to third party
android.permission.ACCESS_COARSE_LOCATION	Access network-based location sources	Allows malicious apps to determine an approximate user's location
android.Permission.WRITE_EXTERNAL_STORAGE	Modify/delete SD card	Allows app to write to the SD card
android.permission.RESTART_PACKAGES	Restart the application	Allows app to be restarted
android.permission.SEND_SMS	Send SMS to undisclosed location	Allows SMS to be sent without user consent
android.permission.READ_LOGS	Read the contents of system's log files	Allows apps to discover information using phone for
android.permission.RECEIVE_SMS	Receive content of user SMS	
android.Permission.INTERNET	Full Internet Access	Allows application to create network sockets
android.permission.READ_PHONE_STATE	Read phone state and Identity	This permission allows app to determine phone number and serial number of a particular phone

C. *Proposed Model Framework*

The existing detection algorithm uses Apriori association analysis for its signature extraction which was characterized with shortcomings. This proposed model used Apriori association analysis that has been improved with Particle Swarm Optimization in order to improve the effectiveness and efficiency of the detection and model performance. The Particle Swarm Optimization is used to generate candidates in the early stage with updated velocity and distance as given in equation (3) and (4). After the candidate generation stage, the Apriori Algorithm is applied to calculate the supports and eventually generate set of best candidate detectors for supervised learning as shown in figure 3.

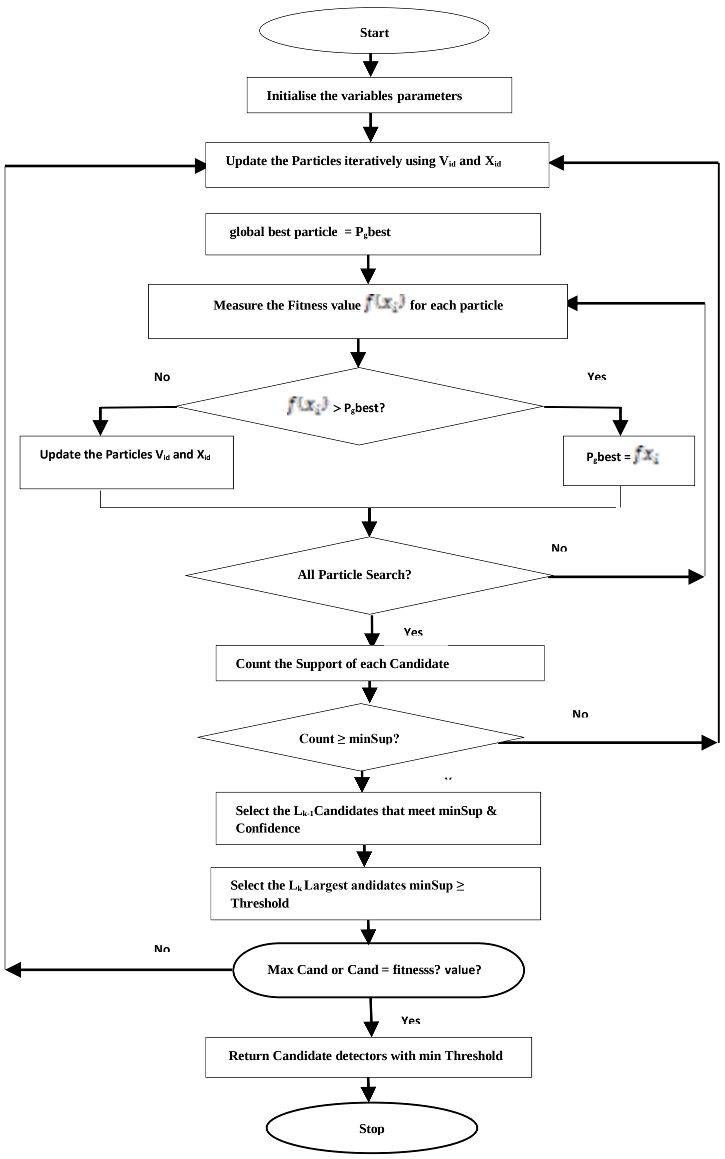

Start
Initialise the variables parameters
Update the Particles iteratively using V_{id} and X_{id}
global best particle = $P_g$best
Measure the Fitness value $f(x_i)$ for each particle
No
Yes
$f(x_i)$ > $P_g$best?
Update the Particles V_{id} and X_{id}
$P_g$best = $f x_i$
No
All Particle Search?
Yes
Count the Support of each Candidate
No
Count ≥ minSup?
Select the L_{k-1}Candidates that meet minSup & Confidence
Select the L_k Largest andidates minSup ≥ Threshold
No
Max Cand or Cand = fitnesss? value?
Yes
Return Candidate detectors with min Threshold
Stop

Figure 3. Proposed Improved AA-Particle Swarm Optimization candidate generation model

D. *Fitness Function*

Negative border otherwise called Atypical factor was used in this research as fitness value to calculate fitness function in order to generate set of acceptable and high ranked features that were otherwise use for model training. Negative border is a set of candidate detectors that are infrequent in the data but whose support is counted. These values increase the efficiency in the generation of large candidate detectors. Orthogonalized Gnanadesikan-Kattenring estimator, OGK estimate [28] was adopted in estimating the distance between the instances of the particle population while an efficient outlier mining algorithm [29] was used in getting the atypical instances called outlier.

The algorithm that used to generate typicality instances and atypical factors is given below:

$Given\ a\ particle\ population\ G = \{G_1, G_2, \ldots G_N \subseteq R^n, where\ Gi\ is\ an\ n\ dimensional\ vector\ with\ velocity\ V_i\ and\ displacement\ D\ of\ every\ i\ particle.$ $The\ velocity, displacement\ and\ attribute\ vector\ R\ that\ correlate\ the\ relation\ between\ the\ instances\ i\ particle\ is\ given\ below\ as\ feature\ dimensional\ vectors$:

$$G_i = \{G_{i1}, G_{i2}, \ldots G_{in}\}^T$$

$$V = \begin{bmatrix} V_{11} & V_{12} \cdots & V_{1n} \\ \vdots & \ldots & \vdots \\ V_{i1} & \cdots & V_{ij} \end{bmatrix} \quad (5)$$

$$where\ V_{ij} = 0\ for\ initial\ velocity\ and\ V_{max}\ for\ maximum\ velocity$$

$$D = \begin{bmatrix} D_{11} & D_{12} \cdots & D_{1n} \\ \vdots & \ldots & \vdots \\ D_{i1} & \cdots & D_{ij} \end{bmatrix} where\ D_{ij}\ is\ the\ euclidean\ distance\ between\ two\ i\ particles\ \in G\ say\ X\ and\ Yand\ denoted\ by\ k = d(X, Y). \quad (6)$$

$Now\ if\ X, Y \in G\ and\ X \neq Y, k\ in\ descending\ order\ for\ particle\ Y, X_1, X_2, X_3 \ldots X_k$ is given as

$$P^k(Y) \quad (7)$$

The sorting of equation (7) in descending order yield a set of typicality scores for every instances. The least integer numbers n that are not lie within the neighbourhood of instances i is called Atypical factors otherwise known as Outliers while the topmost integers for other instances are the candidate detectors otherwise called class prototypes [1].

$$Attribute\ relationR = r_{ij}$$

$$where\ r_{ij}\ is\ a\ vector\ in\ (-1, 1)$$

Note: The neighbourhood of k is denoted by $T^k(Y_i)$, if $N \in T^k(Y_i) >$ k then $F, first\ node \in T^k(Y_i) = 0$ else $T^k(Y_i) = T^k(X_j) = 0$
$where\ T^k(X_j)\ is\ the\ distance\ of\ T^k(Y_i)\ furthest\ kth\ neighbour.$
The algorithm is described by figure 4.

```
Input: G // Random Particle Population
        $i$ // $i$ particle in G
$Output$: $Atypical\ factors\ for\ all\ instances\ of\ particle\ i$
1 Start
2       Initialize Random Particle $i$
3       Initialize distance k, $P^k(Y_i) = \infty, T^k(Y_i) = \emptyset$
            // $set\ of\ Y_i\ k\ nearest\ neighbour$
4       Node $N_i$ = Set of elements in the first Node F
5       If $N_i = leaf\ node\ then$ 6 $Else$ 10
6       $Q$ = d $(Y_i, Y_j)$
7       If d $(Y_i, Y_j)$ < k then $T^k(Y_i) = Y_i$
8       If $|Y_i| > k, then\ F \in T^k(Y_i) = 0\ Else$
9       If $|T^k(Y_i) = k|\ then\ P^k(Y_i) = Q$
10      If $Q \leq \min(P^k), return\ P^k(Y_i)\ Else$ 7
        End If
11      Sort sub-node in descending order
12      Initialize $A = \emptyset$ & $\min(P^k) = 0$
                // $Atypical\ instanses$
13      Construct L-Tree for each $Y_i$ Goto 1
14      If $T^k(Y_i) > \min(P^k)\ then\ A = Y_i$ // $(Y_i \in A)$
15      If |A| > m then minimal $(P^k) = \min(P^k)$
        $Goto$ 7 $Else$ Return A
        End If
16      End
```

Figure 4. Algorithm for Outliers

IV. Empirical Study, Results and Conclusion

This research acquired malware and clean programs from contagiominidump [26] and Googleplay [25] respectively to carry out empirical study. Stratified sampling technique was used to create training and test dataset for better representation for Apriori Algorithm and Apriori-PSO model. The dataset was partitioned into 70% training and 30% test data. Both training and test set were set of .apk files collected as described above. The training data was used to train the model while the test set was used to test the performance of the model. The entire empirical process was discussed in the following subsection.

A. Dataset Analysis

The steps in the empirical process include data collection, program analysis and disassembling, parsing, features extraction, feature selection, independent test on the dataset, and classification model building. Set of Android .apk files were collected for both clean and malicious programs. The programs were made up of 1000 malware from contagiominidump and 500 clean programs from official android market googleplay represents 66.7% and 33.3% respectively. In order to analyze the dataset, static analysis in [22], [23] was adopted using combination of tools. After this initial experiment, we were able to access the source code of the program and useful features were collected.

File analysis was carried out using stratified sampling technique on the entire programs to balance the number of extracted features from malware and clean programs. After the partitioning of the data, each file is parsed and a vector equivalent to each file was extracted as feature. In order to extract best features from the disassembled parsed files, frequent instruction sequences were search globally in the entire data collection using the combined Apriori and PSO algorithm.

Due to the large number of features extracted, which might become redundant to the system, unnecessary features were removed leaving us with moderate features. Statistical test was carried out on the features to examine the existence of relationship or otherwise on the feature and final class value. Those features that were not shown any significant relationship with the target variable were removed from the dataset. The final dataset was represented using a vector space model where each program was a vector in N dimensional point with n number of selected features. A binary variable was defined to represent a malicious application, good application and target variable (malware or benign application).

The combined model extracted rules from set features for a subsequent supervised learning. The mining was done using a 5% support on the partitions which yields separate rules for malware and clean dataset of 650 rules and 350 rules respectively. The combined rules generated from both malware and clean programs are 335 rules. In order to select the best rules from the entire set of rules, a rule found only in a single class was defined and removed in order free the detector of isolated rule. Two percent threshold (2%) was set to identify common rules by calculating the distance in the support level of each class. After the removal of the signature rule and rules common to both classes, the remaining final rules were 325, which denote the frequent features in the collected programs. These rules were presented to the classifiers for supervised learning on which the models were built to classify programs into malware or benign.

B. Criteria for performance evaluation

The criteria for measuring the performance of the proposed method were based on two basic research questions and were done through the use of statistical quality measures usually used in machine learning.

I). Research Questions

The two research questions on which the proposed model was evaluated are:

a) Can we improve the detection rate by train the supervised learners with unsupervised learners rather than using only supervised learners for classification?

b) Is the detection rate of model depends on the quality and quantity of extracted features, feature extraction and selection techniques?

II). *Statistical Test:*

The statistical tests used to evaluate the performance of Apriori association rules and Apriori-PSO in the detection of malware includes Accuracy (ACC), Correlation Coefficient, True positive rate (which measure sensitivity), False positive rate (specificity measure) and Average mean value.

The Accuracy measure

In order to measure the accuracy, we formulate a confusion matrix table represented by figure upon which the accuracy definition was based.

	True Accept (P)	False Reject (N)
True (T)	TP	TN
False (F)	FP	FN

Figure 2. Truth table for Application classification

We defined **TP** (True positive) as the malware that was actually classified as malware i.e. **TPR** is the proportion of positive instances classified correctly.

TN: Benign program that was classified as Benign i.e. **TNR** is the proportion of negative instances classified correctly.

FP: Non-malware that was classified as malware i.e. **FPR** is the proportion of negative instances classified wrongly as positive (malware).

FN: Malware that was classified as Benign i.e. **FNR** is the proportion of positive instances wrongly classified as negative (non-malware).

Therefore:

$$TPR = \frac{TP}{TP+FN} \quad (5)$$

$$TNR = \frac{TN}{TN+FP} \quad (6)$$

$$FPR = \frac{FP}{FP+TN} \quad (7)$$

$$FNR = \frac{TN}{TN+FN} \quad (8)$$

The accuracy actually measures the proportion of correctly classified instances (features)

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (9)$$

Correlation Coefficient (CC) measures the quality of two or more classification techniques in machine learning.

$$CC = \frac{(TP)(TN)-(FP)(FN)}{(TP+FN)\ (TP+FP)(TN+FN)} \quad (10)$$

V. Experimental settings and implementation

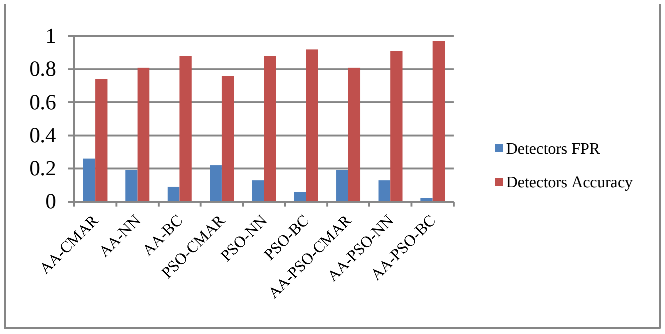
The basis of our experiment was based on research questions defined in section four upon which statistical tests were carried out. First, we aim to compare the effectiveness of combination of supervised with unsupervised learners with using individual classifier for detection. Second is to examine whether the detection rate of model depends on the quality of extracted features, feature extraction and selection techniques. To this end, features were extracted and selected using PSO and Apriori-PSO extraction and selection techniques, three classifiers adopted for classification are CMAR (classification-based Multiple Association Rule), NN (Neural Network), and Bayes classifiers (BC).

Since the accuracy (ACC), false positive rate (FPR), and true positive rate depend on the quality of features and classifier and measure the effectiveness of classifiers, the results display in table 2 and figure 5 and figure 6 obtained as a result of combination of three classifiers with selectors AA, PSO, and AA-PSO over a number of iterations as given below:

a) AA with the three classifiers (NN, CMAR, BC)
b) PSO with the three classifiers (NN, CMAR, BC)
c) AA-PSO with the three classifiers (NN, CMAR, BC)

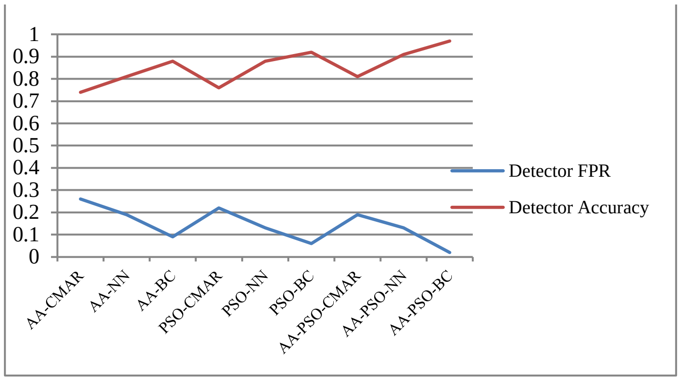
Table 2. Combination of Classifiers with Feature Selectors with three different iterations.

Classifier /Selector	**Iterations**			**Mean FPR**				**Mean Acc**
	100	200	400					
	FPR				ACC			
AA-CMAR	0.446	0.219	0.099	0.255	0.55	0.776	0.893	0.740
AA- NN	0.321	0.159	0.072	0.188	0.676	0.838	0.923	0.812
AA-BC	0.163	0.076	0.039	0.092	0.795	0.898	0.945	0.881
PSO-CMAR	0.382	0.189	0.094	0.222	0.614	0.807	0.890	0.775
PSO-NN	0.222	0.113	0.057	0.131	0.786	0.893	0.947	0.875
PSO-BC	0.097	0.046	0.022	0.055	0.857	0.929	0.964	0.917
AA-PSO-CMAR	0.320	0.159	0.079	0.186	0.676	0.838	0.919	0.811
AA-PSO-NN	0.216	0.117	0.061	0.132	0.845	0.921	0.960	0.909
AA-PSO-BC	**0.030**	**0.015**	**0.007**	**0.017**	**0.952**	**0.976**	**0.988**	**0.972**



Classifiers with selectors

Figure 5. Accuracy and FPR of Selectors with Classifiers



Classifiers with selectors

Figure 6. Accuracy and FPR of Selectors with Classifiers

Figure 5 and 6 shows FPR and Accuracy of combination of Detectors AA, PSO, AA-PSO and classifiers NN, CMAR, BC

Table 3. FPR, TPR, CC, and Accuracy for each combination of Highest Ranked Features and Feature Selectors.

Metrics	**Feature Quantity**	**Selection Methods**		
		Apriori (AA)	**PSO**	**AA-PSO**
FPR	**100**	0.3636	0.1765	0.0790
	200	0.1600	0.0790	0.0375
	400	0.0828	0.0375	0.0183
	700	0.0443	0.0210	0.0104
	1000	**0.0302**	**0.0146**	**0.0072**
TPR				
	100	0.5882	0.7200	0.8478
	200	0.7742	0.8478	0.9205
	400	0.8814	0.8526	0.9593
	700	0.9293	0.9216	0.9765
	1000	**0.9504**	**0.9540**	**0.9835**
Accuracy				
	100	0.6071	0.7619	0.8810
	200	0.8036	0.8810	0.8830
	400	0.8975	0.9405	0.9410
	700	0.9419	0.9660	0.9828
	1000	**0.9598**	**0.9762**	**0.9881**
CC				
	100	0.2395	0.5314	0.7630
	200	0.6106	0.7629	0.8811
	400	0.7274	0.8811	0.9405
	700	0.8243	0.9320	0.9660
	1000	**0.8761**	**0.9524**	**0.9762**

Average Accuracy

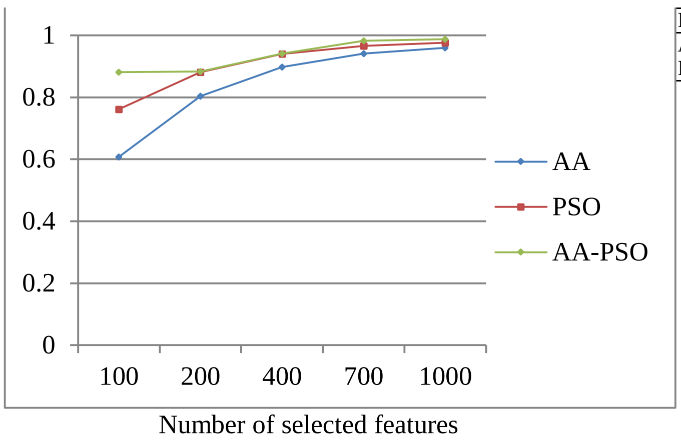


Figure 7. The accuracy of Feature selectors with varying number of features.

VI. Experimental Results and Discussion

In order to compare the effectiveness of an improved AA-PSO, Mean Accuracy and False positive rate of the obtained results were computed to examine the distribution of the populations of the experimented algorithms. It was discovered, at the end of 1000 iteration with threshold values of between 0.1 and 1 that the combination of Apriori and Particle Swarm Optimization (AA-PSO) performance is better than that of AA and PSO. Mean Accuracy, Error rate, and Mean Absolute Error value were also calculated to determine best combination of classifiers and selectors.

Table 4 presents the distribution of AA, PSO, and AA-PSO and shows that there is correlation between means of the three algorithms. Table 4 also shows that the accuracy of AA-PSO is 93.5% compare to that of AA and PSO which stand at 84.2% and 90.5% respectively at 0.2 threshold. The true positive rate and false positive rate of an improved model AA-PSO are 93.8% and 3.1% compare to that of AA and PSO which were 82.5%, 13.6% and 85.9%, 6.6% respectively. The accuracy of AA, PSO, and AA-PSO was further illustrated by the figure 7 which shows that the average accuracy of AA-PSO is better than that of AA and PSO.

Table 5 is used to present the results of the combination of classifiers with selectors. The table shows the best mean accuracy of 97.2% for new model AA-PSO with Bayes classifier over PSO-BC and AA-PSO-NN with 91.7% and 90.9% which follow respectively.

Table 4. Average Values of Model Results for AA, PSO, and AA-PSO.

Model	ACC	CC	FPR	TPR	TP	FP	TN	FN
AA	0.8420	0.6556	0.1362	0.8247	0.8792	0.1139	0.7679	0.1993
PSO	0.9051	0.8120	0.0657	0.8592	0.9431	0.0569	0.8671	0.1329
AA-PSO	**0.9352**	**0.9053**	**0.0305**	**0.9375**	**0.9715**	**0.0285**	**0.9336**	**0.0664**

Table 5. Mean Accuracy, Error Rate, Mean Absolute Error, and Mean Square Error of Three Iterations, 100, 200, and 400.

Model	Mean Acc	Error	MAE
AA-CMAR	0.740	0.260	0.260
AA-NN	0.812	0.188	0.188
AA-BC	0.881	0.119	0.119
PSO-CMAR	0.775	0.225	0.225
PSO-NN	0.875	0.125	0.125
PSO-BC	0.917	0.083	0.083
AA-PSO-CMAR	0.811	0.189	0.189
AA-PSO-NN	0.909	0.092	0.092
AA-PSO-BC	**0.972**	**0.028**	**0.028**

VII. Conclusion

The improvement of the Apriori Algorithm for the extraction and selection of candidate detector for the training of classifiers was explored in this research. The Apriori Algorithm was improved using Particle Swarm Optimization to increase the effectiveness in the generation of candidate detectors for supervised learning. The Atypical variable which represents the instance that does not relate nor has similarity with other instances in the data are used as values to derive fitness function.

In order to test an improved algorithm, permission-based features were extracted from Android application .apk files. The features were used for the classification process of Android applications into malware or benign application. The results of the experimentation, using 1500 malicious and good application from contagiomobile and google play show that an improved model AA-PSO with Bayesian classifier has the best accuracy of 97.2%. The results of FPR and TPR from the experiment also justify the performance of the models through correlation coefficient.

This research combines the supervised and unsupervised learning strategies in order to ensure maximum result in the classification efficiency. The research shows that the static features of a mobile application can be used together with machine learning classifiers through the combination of supervised and unsupervised strategies to classify malicious and good applications. The improved AA-PSO was used as unsupervised strategy to generate candidates that were used to train three different supervised classifiers namely Neural Network, Classification-based Multiple Association Rule (CMAR) and Bayesian classifier (BC). The results supervised classifiers show that the combination of AA-PSO with Bayes Classifier outperforms other two combinations while Neural Network combination with selectors is better than CMAR combination as shown by their mean accuracies and error rates in table 5.

Future Research

The researchers intend to implement this result on an Android smartphone in order to examine the real life efficiency and effectiveness of the improved system.

Acknowledgement

The authors wish to acknowledge the support of International Islamic University Malaysia under the grant research number EDW B14-123-1008.

References

[1] R. J. Roiger and M. W. Geatz, "Data Mining: A Tutorial-Based Primer," Pearson Education Inc. ISBN: 0-201-74128-8, 2003.
[2] R. Agrawal, & R. Srikant, "Fast algorithms for mining association rules." In Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, pp. 487-499, September 1994.
[3] J. Gu, B. Wang, F. Zhang, W. Wang, and M. Gao, "An Improved Apriori Algorithm." In Applied Informatics and Communication, Springer Berlin Heidelberg, pp. 127-133, 2011.
[4] O. S Adebayo, M. A. Mabayoje, A. Mishra, and O. Osho, "Malware Detection, Supportive Software Agents and Its Classification Schemes," International Journal of Network Security & Its Applications (IJNSA), Vol.4 (6), pp. 33 – 49, 2012.
[5] M. A. Siddiqui, "Data Mining Methods for Malware Detection," A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Modeling and Simulation in the College of Sciences at the University of Central Florida, Orlando, Florida, 2008.
[6] B. Abhijit, H. Xin, G. S. Kang and P. Taejoon, "Behavioral detection of Malware on Mobile Handsets," June 17–20, 2008, Breckenridge, Colorado, USA. ACM 978-1- 60558-139-2/08/06, 2008.
[7] I. Idris, A. Selamat, "Improved email spam detection model with negative selection algorithm and particle swarm optimization," Elsevier: Applied Soft Computing, volume 22, pp.11 – 24, 2014.
[8] W. Wang, P. Zhang, Y. Tan, and X. He, "An immune local concentration based virus detection approach." Journal of Zhejiang University Science, pp. 443-454, 2011.
[9] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent PE-malware Detection System Based on Association Mining," Journal in computer virology, 4(4), pp. 323-334, 2008.
[10] H. Wang, X. Z. Gao, X. Huang, and Z. Song, "PSO-optimized negative selection algorithm for anomaly detection." Applications of Soft Computing. Springer Berlin Heidelberg, pp.13-21, 2009.
[11] M. Ohrui, H. Kikuchi, M. Terada, and N. R. Rosyid, "Apriori-PrefixSpan Hybrid Approach for Automated Detection of Botnet Coordinated Attacks. Network-Based Information Systems (NBiS)", 2011 14th International Conference on, IEEE, 2011.
[12] S. S. Garasia, D. P. Rana, and R. G. Mehta, "HTTP Botnet Detection using Frequent Patternset Mining." Proceedings of [Ijesat] International Journal of Engineering Science & Advanced Technology 2: pp. 619-624, 2012.
[13] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization." Pattern Recognition Letters, 28(4), pp. 459-471, 2007.
[14] Y. Tan, "Particle Swarm Optimization Algorithms Inspired by Immunity-Clonal Mechanism and Their Applications to Spam Detection." International Journal of Swarm Intelligence Research (IJSIR), 1(1), pp. 64-86, 2010.
[15] S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, "Particle Swarm Optimization for Parameter Determination and Feature Selection of Support Vector Machines." Expert Systems with Applications, 35(4), pp. 1817-1824, 2008.
[16] C. Bae, W. C. Yeh, Y. Y. Chung, and S. L. Liu, "Feature Selection with Intelligent Dynamic Swarm and Rough Set." Expert Systems with Applications, 37(10), pp. 7026-7032, 2010.
[17] Z. Yi, and Z. Li-Jun, "A rule generation model using s-pso for misuse intrusion detection. In Computer Application and System Modeling (ICCASM)," 2010 International Conference on (Vol. 3, pp. V3-418). IEEE, October 2010.
[18] M. Sheikhan, and M. S. Rad, "Gravitational Search Algorithm–Optimized Neural Misuse Detector with Selected Features by Fuzzy Grids–based Association Rules Mining." Neural Computing and Applications, 23(7-8), pp. 2451-2463, 2013.
[19] Y. Li, and Y. A. Wang, "A Misuse Intrusion Detection Model Based on Hybrid Classifier Algorithm." International Journal of Digital Content Technology and its Applications, Advanced Institute of Convergence Information Technology, 6(5), pp. 25-33, 2012.
[20] S. L. Rosa, S. M. Shamsuddin, and E. Evizal, "An Immune Based Patient Anomaly Detection using RFID Technology." Computer Engineering and Applications Journal, 2(1), pp. 121-142, 2013.
[21] H. Wang, X. Z. Gao, X. Huang, and Z. Song, "PSO-optimized Negative Selection Algorithm for Anomaly Detection." In Applications of Soft Computing, Springer Berlin Heidelberg, pp. 13-21, 2009.
[22] O. S. Adebayo and N. Abdul Aziz, "Techniques for the Analysis of Android Malware." International Conference on Information and Communication Technology For The Muslims World (ICT4M) 2014, Kuching, Sarawak, Malaysia, November, 2014.
[23] V. J. Varghese and S. Walker, "Dissecting Andro Malware," SAN Institute, School of Computer and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK, 2011
[24] M. Kantardzic, "Data Mining: Concepts, Models, Methods, and Algorithms," IEEE Press, ISBN: QA76.9.D343K36 2011, 006.3'12D-dc22, USA, 2011.
[25] Google play. Available at https://play.google.com/store, 2013.
[26] Contagio Mobile. Available at http://www.contagiominidump.com
[27] A. Shabtai, Y. Fledel, & Y. Elovici, "Automated Static Code Analysis for Classifying Android Applications using Machine Learning." International Conference on Computational Intelligence and Security (CIS), 2010.
[28] R. A. Maronna and R. H. Zamar, "Robust Estimates of Location and Dispersion for High-Dimensional Datasets," Technometrics, Vol. 44, No. 4, pp. 307-317, November, 2002. Available at http://www.jstor.org/stable/1271538.
[29] P. Yang and B. Huang, "An Efficient Outlier Mining Algorithm for Large Dataset," 2008 International Conference on Information Management, Innovation Management and Industrial Engineering, pp. 199 – 202, 2008.
[30] M. Schultz, E. Eskin, E. Zadok, S. Stolfo, "Data mining methods for detection of new malicious executables. Proc. IEEE Symposium on Security and Privacy, 2001.

[31] T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan, "N-gram Based Detection of New Malicious Code," Proc. Annual International Computer Software and Applications Conference, 2004.
[32] J.Z. Kolter, M.A. Maloof, "Learning to detect malicious executables in the wild," Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 470–478.
[33] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, Y. Elovici, "Unknown Malcode Detection via Text Categorization and the Imbalance Problem," Proc. IEEE Intelligence and Security Informatics, Taiwan, 2008.
[34] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory," In Proceedings of the sixth international symposium on micro machine and human science, Vol. 1, pp. 39-43, October 1995.
[35] P. Yang and B. Huang, "An Efficient Outlier Mining Algorithm for Large Dataset," 2008 International Conference on Information Management, Innovation Management and Industrial Engineering, pp. 199 – 202, 2008.
[36] O. S. Adebayo and N. Abdul Aziz "Android Malware Classification Using Static Code Analysis and Apriori Algorithm Improved with Particle Swarm Optimization" 4th World Congress on Information and Communication Technologies Malacca, Malaysia December 08–10, 2014.
[37] J. J. Drake, P. O. Fora, Z. Lanier, C. Mulliner, S. A. Ridley, & G. Wicherski (2014). Android Hacker's Handbook, United State of America, Indianapolis, Indiana: John Willey & Sons, Incorporation.
[38] Y. Shi, and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization". In Evolutionary Computation," Proceedings of the 2001 Congress of IEEE, Vol. 1, pp. 101-106, 2001.
[39] M. S. Abadeh, J. Habibi, S. and Aliari, "Using a Particle Swarm Optimization Approach for Evolutionary Fuzzy Rule Learning: A Case Study of Intrusion Detection," In Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU), pp. 2-7, July 2006.
[40] X. Hu, and R. Eberhart, "Multi-objective Optimization Using Dynamic Neighborhood Particle Swarm Optimization, In Computational Intelligence," Proceedings of the World on Congress on IEEE, vol. 2, pp. 1677-1681, May 2002.
[41] R. Siddiqui, and S. Khatibi, "Visual Tracking using Particle Swarm Optimization," arXiv preprint arXiv:1401.4648, 2014.
[42] M. Clerc, and J. Kennedy, "The Particle Swarm-explosion, Stability, and Convergence in a Multidimensional Complex space, Evolutionary Computation," IEEE Transactions, 6(1), 58-73, 2002.
[43] Y. Tan, and Z. M. Xiao, "Clonal Particle Swarm Optimization and its Applications," In Evolutionary Computation, 2007, CEC 2007. IEEE Congress, pp. 2303-2309, September 2007.
[44] Y. Shi, and R. Eberhart, "A Modified Particle Swarm Optimizer," In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, the 1998 IEEE International Conference, pp.69-73, May 19.
[45] Aafer, Yousra, Wenliang Du, and Heng Yin. "DroidAPIMiner: Mining API-level features for robust malware detection in android." *Security and Privacy in Communication Networks*. Springer International Publishing, 2013. 86-103.
[46] Christina Warrender, Forrest, S. & Barak, A. "Pearlmutter Detecting intrusions using system calls: Alternative data models", In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.
[47] Christodorescu, M., Jha, S., Seshia, S.A., Song, D. and R.E.Bryant. "Semantics-aware malware detection", In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.
[48] Lee, T. and Mody, J.J. (2006) Behavioral Classification. Proceedings of the European Institute for Computer Antivirus Research Conference (EICAR'06).
[49] Thomas Eder, Michael Rodler, Dieter Vymazal, Markus Zeilinger "A Framework For Analyzing Android Applications". Workshop on Emerging Cyberthreats and Countermeasures ECTCM 2013.
[50] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, Yael Weiss "Andromaly": a behavioral malware detection framework for android devices**".** *Journal of Intelligent Information Systems* 38(1) (January 2011) 161{190}, 2011.
[51] Iker Burguera, Urko Zurutuza, Simin Nadjm-Tehrani "Crowdroid: Behavior-Based Malware Detection System for Android". In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and mobile devices (October 2011), Pp.15-26, 2011.
[52] Gianluca Dini, Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra "MADAM: a Multi-Level Anomaly Detector for Android Malware" MMM-ACNS'12 In proceedings of 6th international conference on Mathematical Methods, Models and Architectures for Computer Network Security: computer security pp. 240-253, 2012.
[53] Abela, Kevin Joshua L., Angeles, Don Kristopher E., Delas Alas, Jan Raynier P., Tolentino, Robert Joseph, Gomez, Miguel Alberto N. "An Automated Malware Detection System for Android using Behavior-based Analysis, AMDA" *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 2(2), 2013: The Society of Digital Information and Wireless Communications, ISSN: 2305-0012.
[54] Thomas Blasing, Leonid Batyuk, Aubrey-Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak (University of Berlin), Malware "An Android Application Sandbox system for suspicious software detection", 2010.
[55] Suhas Holla, and Mahima M Katti "Android based Mobile Application and Its Security". International Journal of Computer Trends and Technology, 3 (3) Pp. 486 – 490, 2013. ISSN 2231 – 2801.
[56] Andrew Walenstein, Luke Deshotels, and Arun Lakhotia "Program Structure-Based Feature Selection for Android Malware Analysis" MOBISEC 2012, LNICST 107, pp. 51–52, 2012. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering

Author Biographies

Olawale Surajudeen Adebayo was born in Oyo State of Nigeria in 1974. Presently a Lecturer in the Department of Cyber Security Science, Federal University of Technology Minna, Niger State, Nigeria and a PhD research student in the Department of Computer Security Science, International Islamic University Malaysia. He bagged Bachelor of Technology in Mathematics and Computer science from Federal University of Technology, Minna, Nigeria in 2004 and the MSc. in Computer science from University of Ilorin, Kwara State, Nigeria in 2009. His current research interests include: Malware Detection, Information Security, Cryptology, and Data Mining Security. He has published many academic papers in the above-mentioned research areas. He is a member of Computer Professional Registration Council of Nigeria (CPN), Nigeria Computer Society (NCS), IEEE, Global Development Network, International Association of Engineers

(IAENG) and many others. He is a reviewer to many local and international journals. See more at http://www.osadebayo.com.

Normaziah A. Aziz, obtained her Bachelor in Computer Science from the University of South Carolina, USA, Masters in Computer Science from Universiti Kebangsaan Malaysia , and later her PhD from the Dept. of Artificial Intelligence, University of Edinburgh, Scotland, UK. She was a Senior Research Fellow and Head for the Knowledge Representation and Reasoning, in the AI lab at MIMOS, an R&D organization in Malaysia. Presently she is an Associate Professor in the Dept. of Computer Science, International Islamic University Malaysia. Her research work and interest are in the areas of Cognitive Modeling, Natural Language Processing, Digital Evidence Forensics and Malware analysis.