SwarmDroid: Swarm Optimized Intrusion Detection System for the Android Mobile Enterprise

Abimbola Adebisi Adigun[1], Temitayo Matthew Fagbola[2] and Adekanmi Adegun[3]

[1] Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria

[2] Department of Computer Science, Federal University, Oye-Ekiti, Ekiti State, Nigeria

[3] Department of Computer Science, Landmark University, Omu-Aran, Kwara State, Nigeria

Abstract

The inadequacies inherent in the current defense mechanism of the mobile enterprise led to the development of new breed of security systems known as mobile intrusion detection system. The major worry of mobile / ubiquitous device users is the issue of data security since no mobile security application is 100% efficient. Existing studies conducted on android mobile security reveal that Android is the platform with the highest malware growth rate by the end of 2011 and that Global System for Mobile Communication -based Pivot Attacks, Mobile Botnets and Malicious Applications are the major security vulnerabilities compromising the confidentiality, integrity and availability of this mobile enterprise. In this paper, a SwarmDroid IDS is developed following a machine learning approach using Support Vector Machine. NSL-KDD dataset was used to test and evaluate the performance of the SwarmDroid IDS and compared with J48 and Random Forest which are state-of-the-art machine learning techniques for intrusion detection in mobiles. Particle Swarm Optimization was used for feature selection. The malware detection systems were simulated in a MATLAB environment. The SwarmDroid IDS was evaluated using detection time, true positive rate, false positive rate and detection accuracy as performance metrics. The result obtained from the evaluation revealed that SwarmDroid IDS outperforms J48 in terms of detection time and accuracy. Also, feature selection in Android application package files using particle swarm optimization technique plays a critical role in realizing high accuracy and low computational time complexity in SwarmDroid.

Keywords: *SwarmDroid, Android, Random Forest, J48.*

1. Introduction

The Modern mobile platforms are reinventing the mobile landscape. These mobile devices run commodity operating systems and have complete multi-protocol networking stacks, user interface toolkits, file systems and other fully-featured libraries (Jon, 2010). While past mobile platforms had limited functionality and were relatively close to third-party applications and user extensibility, new mobile platforms are now being deployed with complex internet, productivity, communication and application suites which strongly encourage third-party development of comprehensive software development kits and application delivery mechanisms (Nohl & Melette, 2011).

However, these mobile devices face a wide range of new security challenges including malicious threats and intrusion (Jon, 2010). The same extensibility that has enabled rich functionality and applications has also made them an enticing target for attackers. These devices are increasingly being used to store sensitive personal information such as financial data used for mobile banking, but also run applications that pose potential abuse for snooping on a mobile user's voice, SMS, data and location services (Jon,Veeraraghavan, Cooke, Flinn and Jahanian, 2008). As such, there is a need for an intrusion prevention support system for the mobile enterprise. Intrusion prevention in a mobile enterprise aims at identifying any entity that attempts to compromise the confidentiality, integrity or availability of a computing resource (SANS, 2002). The android platform, an open-source mobile operating system, is susceptible to vulnerabilities such as GSM based Pivot Attacks, mobile botnets, Malicious Applications, Infection via Personal Computers, Device to device Infection and Infection via Rogue Wireless Networks (AISEC, 2012; Nohl & Melette, 2011; Jon, 2010).

Existing studies conducted on android mobile security reveal that Android is the platform with the highest malware growth rate by the end of 2011 (Zarni and Win, 2013). To deal with these shortcomings, malware detection profiles should be updated with a large amount of the newest engine and malware definitions at regular interval of time (Mark and Smith, 2007). Consequentially, a large amount of the engine definitions also increases the problem of inconsistency, redundancy and ambiguity and thus a need for optimal definition selection. In this paper, SwarmDroid, a computationally-efficient swarm-optimized android intrusion detection system (IDS) is developed to address the major security vulnerabilities infiltrating the android mobile platform, so as to realize a more secured and reliable android operating environment.

2. Literature Review

This section introduces a brief literature, conceptual framework and major security challenges of the Android mobile enterprise.

The Android Platform
Mobile operating systems preinstalled on all currently sold smart phones need to meet different criteria than desktop and server operating systems, both in functionality and security. Mobile platforms often contain strongly interconnected, small and less well controlled applications from various single developers, whereas desktop and server platforms obtain largely independent software from trusted sources (Ekberg & Kyl, 2007). Also, users typically have full access to administrative functions on non-mobile platforms. Mobile platforms, however, restrict administrative control through users. As a consequence, different approaches are deployed by the Android platform to maintain security. Applications for Android are developed in Java and executed in a virtual machine, called Dalvik VM (Jon, 2010). They are supported by the application framework, which provides frequently used functionality through a unified interface. Various libraries enable applications to implement graphics, encrypted communication or databases easily (AISEC, 2012). The Standard Library ("bionic") is a BSD derived library for embedded devices. The respective Android releases" kernels are stripped down from Linux 2.6 versions. Basic services such as memory, process and user management are all provided by the Linux kernel in a mostly unmodified form (AISEC, 2012). The Android system architecture is presented in figure 1 below. However, for several Android versions, the deployed kernel's version was already out of date at the time of release. This has led to a strong increase in vulnerability (Wenjia & Anupam, 2003), as exploits were long publicly available before the respective Android version's release.

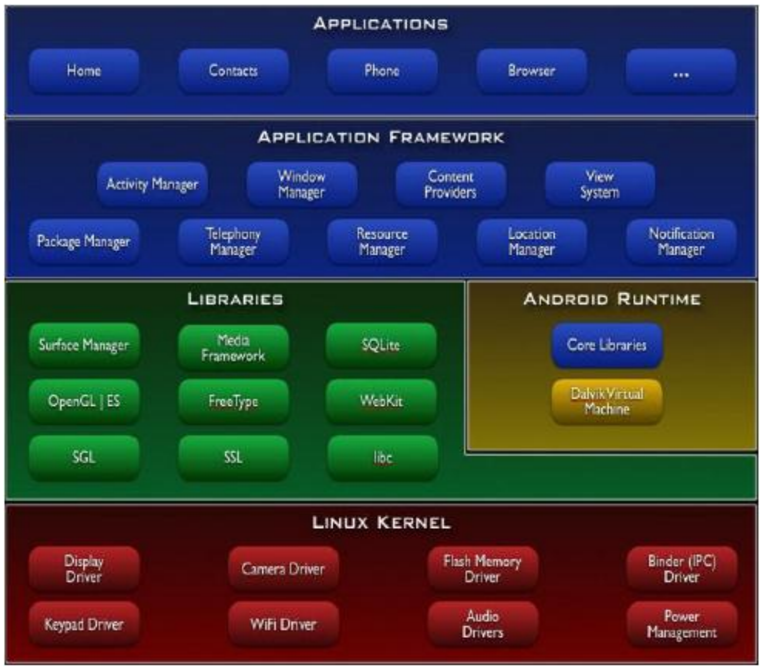

Fig. 1 Android System Architecture. Source: AISEC (2012)

Security Challenges in Android Mobile Network
Mobile environments differ significantly from traditional fixed computing environments. While some differences are straightforward, others may have subtle consequences that can have a significant impact on the security of a mobile device. The major security vulnerabilities of the android enterprise addressed in this paper include mobile botnets, GSM-based pivot attack and malicious applications.

A. Mobile Botnets: Compared to traditional fixed computing, the case for mass ownage of mobile devices for creating a botnet is not as straightforward. Traditionally, attackers are able to monetize their botnets of compromised hosts through spam, denial of service extortion, sensitive data theft and phishing of confidential details (Dagon and Starner, 2004). Compromised hosts are often considered valuable to an attacker if they have high-throughput, low-latency, stable connectivity to the Internet and significant system resources; which are attributes that are not common with today's mobile devices (Ekberg & Kyl, 2007). However, as more and more sensitive data such as login credentials are stored on mobile devices, attackers may still wish to target them for harvesting data.

B. GSM based Pivot Attacks: The GSM implementation by many base transceiver stations is prone to various easily conductible attacks. Recent results presented at the 28th Chaos Communication Congress have demonstrated that most European GSM networks are not capable of prohibiting impersonation attacks (Brunner, Hofinger, Krauss, Roblee, Schoo and Todt, 2010). This occurs when an attacker fakes the identity of another GSM subscriber, thus receiving any communication addressed to the attacked person (Brunner et al, 2010). Hence, just one mobile device under an attacker‟s control in a radio cell is sufficient to attack any other subscriber and to serve as a remote long range wiretap. This matter, though theoretically feasible, is of very high difficulty and no further research into it has been conducted yet (Brunner et al, 2010).

C. Malicious Applications: Malicious applications are software used or created to disrupt computer operation, gather sensitive information or gain access to private computer systems and mobile devices (Nohl & Melette, 2011). Incidents have many causes, such as malware (worms & spyware), attackers gaining unauthorized access to systems from the Internet and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized. Although many incidents are malicious in nature, many others are not; for example, a person might mistype the address of a computer and accidentally attempt to connect to a different system without authorization (Bace & Rebecca, 2000). Some Intrusion Prevention System (IPS) technologies can remove or replace malicious portions of an attack to make it benign.

3. Review of Related Works

Zarni and Win (2013) proposed a framework for machine learning-based malware detection system on Android to detect malware applications and to enhance security and privacy of smartphone users. This system monitors various permission-based features and events obtained from the android applications and analyze these features by using machine learning classifiers to classify whether the application is a goodware or malware.
Sven and Stephan (2013) presented the design and implementation of FlaskDroid, a policy-driven generic two-layer MAC framework on Android-based platforms. They introduced an efficient policy language that is tailored for Android‟s middleware semantics. The applicability of the design was prototyped on Android 4.0.4. Evaluation of the system shows that the clear API-oriented design of Android benefits the effective and efficient implementation of a generic mandatory access control framework like FlaskDroid.
Rafael, Julian and Marcel (2013) evaluated how well Android antivirus software performs under real world conditions, as opposed to retrospective detection rate tests. The authors conducted various tests on several antivirus apps for Android. The test setup considers the ability to cope with typical malware distribution channels, infection routines and privilege escalation techniques. It was concluded that it is easy for malware to evade detection by most antivirus apps with only trivial alterations to their package files.
Shabtai, Kanonov, Elovici, Glezer and Weiss (2011) developed a malware detection framework for android devices and tagged it Andromaly. The framework monitors both the behavior of Android users using eighty-eight (88) features via several parameters, ranging from CPU usage to sensors activities by selecting the features that describe users‟ behavior and pre-processed using feature selection algorithms.
Dini, Martinelli, Saracino and Sgandurra (2012) proposed a Multi-Level Anomaly Detector for Android Malware (MADAM) which uses thirteen (13) features to detect android malware for both user level and kernel level. The application was tested on real malware and uses a global-monitoring approach that can detect malware contained in unknown applications.
Karen (2009) developed an IDPS that could report mobile attacks and security breaches to security administrators, who could quickly initiate incident response actions to minimize the damage caused by the incident.
Kruegel & Chris (2004) developed a malware detector that can identify reconnaissance activity in Android, which may indicate that an attack is imminent. For example, some attack tools and forms of malware, particularly worms, perform reconnaissance activities such as host and port scans to identify targets for subsequent attacks. The IDPS is able to block reconnaissance and notify security administrators, who can take actions if needed to alter other security controls to prevent related incidents. Because reconnaissance activity is so frequent on the Internet, reconnaissance detection is often performed primarily on protected internal networks.

4. Materials and Method

The detailed methodology and design approach adopted are described as follow:

A. Support Vector Machine

In machine learning, Support Vector Machines (SVM) are learning models with learning algorithm that analyze data and recognize patterns, used for classification (Cortes et al., 1995). Classification is the problem of identifying to which of a set of categories (sub-population) a new observations (or instances) whose category membership is known. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output. Given a set of training samples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new samples into one category or the other. It represents these samples as point in space mapped so that the samples of the separate categories are divided by a clear gap that is as wide as possible.

New samples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. SVM constructs a hyper-plane in a high dimension space which functions as a separating plane for classification. Points with different class labels are separated by the hyper-plane while those with the same class labels are kept in the same partition.

There are two classes (Fagbola et al., 2012):

$$y_i \in \{-1, 1\} \quad (1)$$

and there are ***N*** labeled training examples:

$$(x_1, y_1), \dots\dots, (x_N, y_N), x \in R^d \quad (2)$$

where *d* is the dimensionality of the vector. If the two classes are linearly separable, then one can find an optimal weight vector W^* such that $\|W^*\|^2$ is minimum; and

$$W^* \bullet x_i - b \geq 1 \ if \ y_i = 1$$
$$W^* \bullet x_i - b \leq -1 \ if \ y_i = -1 \quad (3)$$

or equivalently

$$y_i(W^* \bullet x_i - b) \geq 1 \quad (4)$$

Training examples that satisfy the equality are termed support vectors. The support vectors define two hyper-planes, one that goes through the support vectors of one class and one goes through the support vectors of the other class. The distance between the two hyper-planes defines a margin and this margin is maximized when the norm of the weight vector $\|W^*\|$ is minimum.

This minimization can be performed by maximizing the following function with respect to the variables α_j

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (x_i \bullet x_j) y_i y_j \quad (5)$$

subject to the constraint: $0 > \alpha_j$ where it is assumed there are ***N*** training examples, x_i is one of the training vectors, and ● represents the dot product. If $\alpha_j > 0$ then x_j is termed a support vector. For an unknown vector x_j classification then corresponds to finding

$$F(x_j) = sign\{W ^{*}\bullet x_j - b\} \quad (6)$$

Where

$$W^* = \sum_{i=1}^{\Upsilon} \alpha_i y_i x_i \quad (7)$$

and the sum is over the ϒ nonzero support vectors (whose α's are nonzero).

The advantage of the linear representation is that W^* can be calculated after training and classification amounts to computing the dot product of this optimum weight vector with the input vector.

For the non-separable case, training errors are allowed and minimizing

$$\|W^*\|^2 + C \sum_{i=1}^{N} \xi_{ij} \quad (8)$$

subject to the constraint

$$y_i(W^* \bullet x_i - b) \leq 1 - \xi \quad \xi \geq 0 \quad (9)$$

Where ξ is a slack variable and allows training examples to exist in the region between the two hyper-planes that go through the support points of the two classes.

B. Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic global optimization technique developed by Eberhart and Kennedy in 1995 based on social behavior of birds (Clerc, 2002). PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the idea is emerged (Kennedy, 2001). PSO performs searches using a population (called swarm). The population consists of potential solutions, named particles, which are a metaphor of birds in flocks. These particles are updated from iteration to iteration.

In PSO, a set of particles or solutions traverse the search space with a velocity based on their own experience and the experience of their neighbors. Each particle updates its own velocity and position based on the best experience of its own and the entire population. This process is repeated till an optimal solution is obtained.

The detailed operation of particle swarm optimization is given below (Karl, 2005):

Step 1: Initialization. The velocity and position of all particles are randomly set to within pre-defined ranges.

Step 2: Velocity Updating. At each iteration, the velocities of all particles are updated according to:

$$V_i = WV_i + C_1R_1(P_{i,best} - P_i) + C_2R_2(g_{i,best} - P_i) \quad (10)$$

where P_i and V_i are the position and velocity of particle *i*, respectively; $P_{i,best}$ and $g_{i,best}$ is the position with the 'best' objective value found so far by particle *i* and the entire population respectively; *w* is a parameter controlling the flying dynamics; R1 and R2 are random variables in the range [0, 1]; c1 and c2 are factors controlling the related weighting of corresponding terms. The inclusion of random variables endows the PSO with the ability of stochastic searching. The weighting factors c1 and c2 compromise the inevitable tradeoff between exploration and exploitation. After updating, V_i should be checked and secured within a pre-specified range to avoid violent random walking.

Step 3: Position Updating. Assuming a unit time interval between successive iterations, the positions of all particles are updated according to:

$$P_1 = P_i + V_i \tag{11}$$

After updating, P_i should be checked and limited to the allowed range.

Step 4: Memory updating. Update $P_{i,best}$ and $g_{i,best}$ when condition is met.

$$P_{i,best} = P_i \quad \text{If } f(P_i) > f(P_{i,best})$$
$$g_{i,best} = g_i \quad \text{If } f(g_i) > f(g_{i,best}) \tag{12}$$

where $f(x)$ is the objective function subject to maximization.

Step 5: Termination Checking. The algorithm repeats Steps 2 to 4 until certain termination conditions are met, such as a pre-defined number of iterations or a failure to make progress for a certain number of iterations. Once terminated, the algorithm reports the values of g_{best} and f (g_{best}) as its solution.

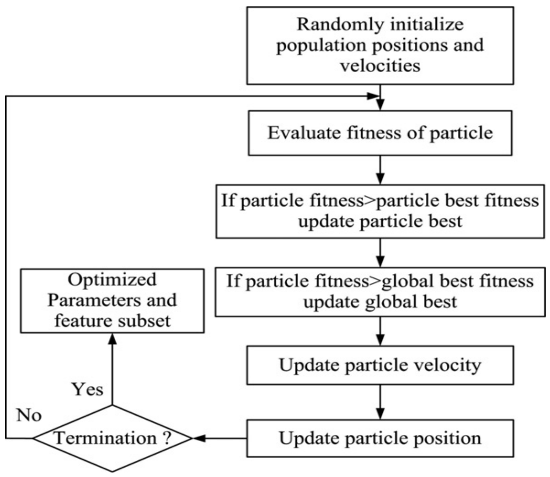
Randomly initialize population positions and velocities
Evaluate fitness of particle
If particle fitness>particle best fitness update particle best
If particle fitness>global best fitness update global best
Update particle velocity
Update particle position
Termination ?
Yes
No
Optimized Parameters and feature subset

Fig. 2 The Flowchart of PSO Algorithm (Lin et al., 2008).

C. J48 Decision Tree Algorithm

Input: training sample set T, the collection of candidate attribute attribute_list
Output: a decision tree.
Steps:

```
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 remainder samples of T is less than a given value, then return N as a leaf node, and mark it as the category which appears most frequently in attribute_list, for each attribute, 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 attribute list:
5. If testing attribute is continuous, then find its division threshold;
6. For each new leaf node grown 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 J48_Tree (T', T'_attributelist)
   }
7. Calculate the classification error rate of each node and then prune the tree.
```

D. The Random Forest Algorithm

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 for 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.

Steps:

```
1. Select ntree, the number of trees to grow, and mtry, a number larger than number of variables.
2. For i = 1 to ntree:
3. Draw a bootstrap sample from the data. Call those not in the bootstrap sample the "out-of-bag" data.
4. Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree is grown to maximum size and not pruned back.
5. Use the tree to predict out-of-bag data.
6. In the end, use the predictions on out-of-bag data to form majority votes.
7. Prediction of test data is done by majority votes from predictions from the ensemble of trees.
```

E. The SwarmDroid Model Design

The SwarmDroid IDS model design treats malware detection as a binary classification problem with Android application package files containing either malware or goodware. For malware detection on Android platform, features were retrieved for each Android application from its corresponding application package (APK) files. However, some of these features were redundant and irrelevant and could make the detection process computationally very expensive. To address this, PSO was used for optimal feature selection to help reduce the computational burdens (memory and CPU time required to detect attack) of the intrusion detection system and also improve the classification accuracy while SVM was used for binary classification of the optimally-selected features.

The feature selection process can be considered a problem of global combinatorial optimization in machine learning, which reduces the number of features, removes irrelevant, noisy and redundant data, and results in acceptable classification accuracy (Fagbola et al., 2012).
Feature subset selection can involve random or systematic selection of inputs or formulation as an optimization problem which involves searching the solution space of feature set for an optimal or near-optimal subset of features, according to a specified criterion.

Conducting feature selection is usually directed toward one of two goals (Fagbola et al., 2012):

1. Ability to minimize the number of features selected while satisfying some minimal level of classification capability

2. Ability to maximize classification performance for a subset of prescribed cardinality. The feature selection process can be improved by optimizing its corresponding feature subset selection technique(s) using some appropriate metaheuristic optimizers.

The main steps of the developed SwarmDroid model are as follows:

1. Feature extraction of APK files.
2. Particle Swarm Optimization to generate and select both the optimal feature subset and SVM parameters at the same time.
3. Classification of the resulting optimal features by SVM.

Firstly, the best feature subset is selected using PSO. Each particle represents a solution, which denotes the selected subset of features and parameter values.

The selected features, parameter values and training dataset are used to train the SVM classifier model as shown in figure 3. If n features are required to decide which features are chosen, then 2 + n decision variables must be adopted (Lin et al., 2008). The value of n variables ranges between 0 and 1. If the value of a variable is less than or equal to 0.5, then its corresponding feature is not chosen. Conversely, if the value of a variable is greater than 0.5, then its corresponding feature is chosen. The developed SwarmDroid IDS architecture is presented in figure 4.

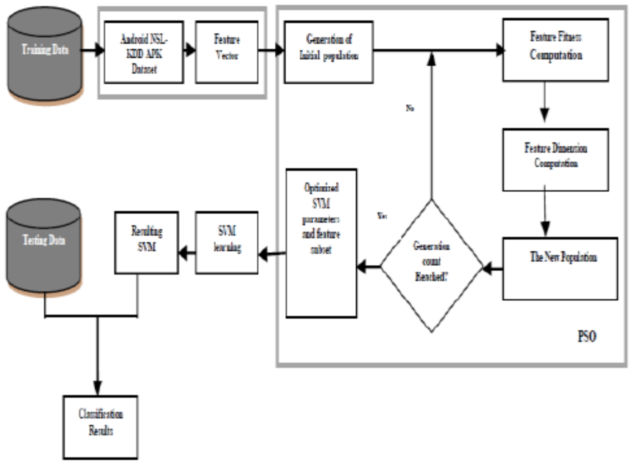
Training Data
Android NSL-KDD APK Dataset
Feature Vector
Generation of Initial population
Feature Fitness Computation
No
Feature Dimension Computation
Optimized SVM parameters and feature subset
Yes
Generation count Reached?
The New Population
Testing Data
Resulting SVM
SVM learning
PSO
Classification Results

Fig. 3 The SwarmDroid IDS Model Design

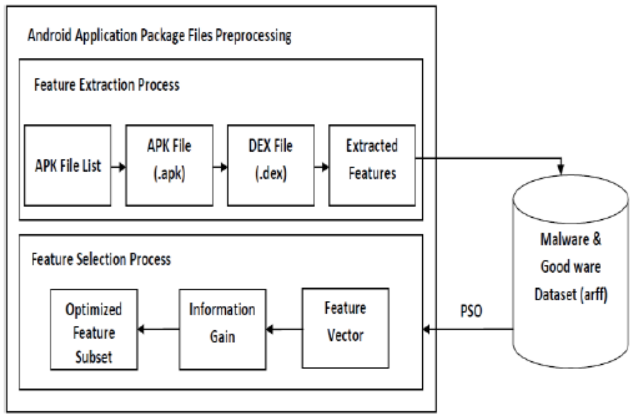
Android Application Package Files Preprocessing
Feature Extraction Process
APK File List
APK File (.apk)
DEX File (.dex)
Extracted Features
Malware & Good ware Dataset (arff)
Feature Selection Process
Optimized Feature Subset
Information Gain
Feature Vector
PSO

Fig. 4 The Developed SwarmDroid IDS Architecture

F. The NSL-KDD Dataset

In this paper, the NSL-KDD, a publicly available dataset (which consists of selected records of the complete KDD data set and does not suffer from uneven distribution of attacks as noticed in KDD) suitable for the evaluation of Intrusion Detection Systems (Debar, 2009), is used to test and evaluate the performance of the developed SwarmDroid IDS and compared with J48 and Random Forest machine learning techniques considered for benchmark purpose. This dataset is characterized by mobile botnets, GSM-based Pivot Attacks and Malicious Application malwares among others which makes it suitable for testing and evaluation purpose in this work.

G. The Performance Evaluation Metrics

Given a particular attack category, the rate of detection by malware detection algorithms differ in performance. Hence, there is a need for performance evaluation of these algorithms. The performance of the proposed swarm optimized technique over the machine learning techniques comparatively considered were evaluated using detection time, true positive rate, false positive rate and detection accuracy as performance evaluation metrics as defined below:

1. **True Positive Rate (TPR)**: Percentage of correctly identified goodware application files. TPR = (TP / TP + FN) where FN is the false negative, the number of wrongly identified goodware applications.
2. **False Positive Rate (FPR)**: Percentage of wrongly identified malware application files where FPR = (FP / FP + TN).
3. **True Negative (TN)**: Number of correctly identified malware application files.
4. **False Negative (FN)**: Number of wrongly identified goodware application files.
5. **Detection Accuracy:** Percentage of correctly identified applications (TP+TN / TP+TN+FP+FN).
6. **Detection Time**: This represents the total time required to process and detect all malware application files.

5. Results and Discussion

The result of evaluation of the developed SwarmDroid is presented in this section:

Figure 5 presents the menu for the four (4) machine learming techniques considered in this paper. These are SVM, J48, Random Forest and SwarmDroid.

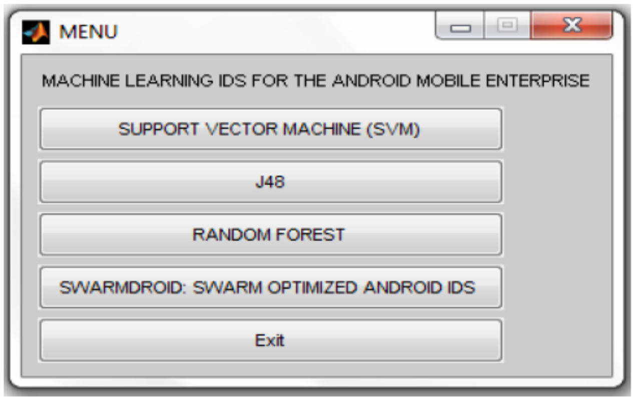
MENU
MACHINE LEARNING IDS FOR THE ANDROID MOBILE ENTERPRISE
SUPPORT VECTOR MACHINE (SVM)
J48
RANDOM FOREST
SWARMDROID: SWARM OPTIMIZED ANDROID IDS
Exit

Fig 5 The Menu for the Machine Learning Technique

In figure 6, the numbers indicate the number of malware detected by each intrusion detection system. J48 shows remarkable improvement over random forest and SVM however, SwarmDroid outperforms SVM, J48 and Random Forest in the detection of mobile botnets, GSM-based Pivot Attacks and Malicious Applications as it scores the highest detection rate.

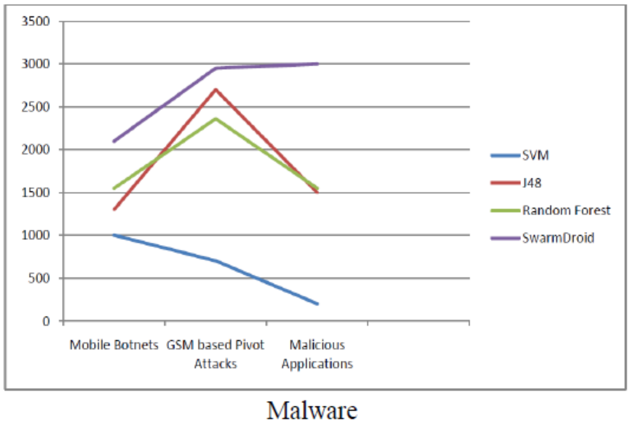

Fig. 6 The Summarized Evaluation Result of SVM, SwarmDroid, J48 and Random Forest

Table 1 presents the summary of the result obtained using SVM and SwarmDroid for malware detection. The evaluation was conducted using NSL-KDD dataset and SVM and SwarmDroid were introduced to varying feature sizes of 100, 300 and 600 Android apk files. Each training set contained 75% of the original set while the test set contained 25% of the original dataset and were randomly selected.

Table 1: SVM-based and SwarmDroid Malware Detection Results

Nos of Features	SVM-based Malware Detection		SwarmDroid Malware Detection	
	Detection Accuracy (%)	DetectionTime (secs)	Detection Accuracy (%)	Detection Time (secs)
100	68.3438	6.3281	80.4375	2.0625
300	46.7062	60.1563	90.5625	0.5625
600	18.015	91.465	93.1937	0.1938

It is observed that SwarmDroid yields a higher detection accuracy of 80.4375%, 90.5625% and 93.1937% within lesser computational times of 2.0625, 0.5625, and 0.1938 seconds while ordinary SVM yields lesser detection accuracy of 68.3438%, 46.7062% and 18.015% with higher computational time of 6.3281, 60.1563 and 91.465 seconds for NSL-KDD feature sizes of 100, 300 and 600 respectively. This result confirms the report of Andrew (2010) and Liyang et al. (2005) that though SVM is much more effective than other conventional non-parametric classifiers in terms of classification accuracy, computational time and stability to parameter setting; it is weak in its attempt to classify highly dimensional dataset with large number of features (Andrew 2010). This implies that the malware detection efficiency of SwarmDroid surpassed that of conventional SVM due to optimal feature subset selection using PSO.

6. Conclusions

Android platform, an open-source operating system, is susceptible to major vulnerabilities such as GSM based Pivot Attacks, Mobile Botnets and Malicious Applications. These challenges necessitate the development of a security support application for the Android enterprise. However, in this paper, SwarmDroid, a swarm optimized Android IDS is developed to address these vulnerabilities. Three (3) state-of-the-art machine learning techniques for intrusion detection were considered.

However, SwarmDroid was found to be the most efficient in terms of detection time and accuracy. This reveals that feature selection of Android APK files using PSO plays a critical role in realizing higher accuracy with minimum computation resource requirement. Future research work can extend the detection of malware in Android to other vulnerabilities not considered in this paper.

References

[1] Bace and Rebecca "*Intrusion Detection*," Macmillan Technical Publishing, 2000.
[2] T. Brunner, H. Hofinger, C. Krauss, C. Roblee, P. Schoo and S. Todt, "Infiltrating Critical Infrastructure with Next Generation Attacks; Stuxnet as a Showcase Threat", 2010.
[3] T. Dagon Martin and T. Starner, "Mobile phones as computing devices: The viruses are coming", IEEE Pervasive Computing", 2004.
[4] K. Ekberg and M. Kyl "Mobile Trusted Module (MTM), An Introduction" Nokia Research, 2007.
[5] AISEC "Android OS Security: Risks and Limitations", AISEC 2012.
[6] Jon Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian "Virtualized In-Cloud Security Services for Mobile Devices in Workshop on Virtualization in Mobile Computing (MobiVirt '08)", Breckenridge, Colorado, 2008.
[7] Jon Oberheide "remote kill and install on google android,". http://jon.oberheide.org/blog/2010/06/25/remotekillandinstallongoogleandroid/, 2010.
[8] Karen Scarfone "Guide to Intrusion Detection and Prevention Systems (IDPS), 2009"
[9] K. Nohl and Melette L. "Defending mobile phones," in *28th Chaos Communication Congress*, http://events.ccc.de/congress/2011/Fahrplan/attachments/1994_111217.SRLabs28C3Defending, 2011.
[10] Rafael Fedler, Julian Schütte, Marcel Kulicke , "On the Effectiveness of Malware Protection on Android, An evaluation of Android antivirus apps", Applied and Integrated Security (2013).
[11] SANS Institute "Intrusion Prevention Systems- Security", 2002.
[12] S. Bugiel and Stephan Heuser, "Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies", The Internet Society (2013).
[13] L. Wenjia and J. Anupam, "Security Issues in Mobile Ad Hoc Networks- A Survey" 2003.
[14] Karl o. Jones, Comparison of Genetic Algorithm and Particle Swarm Optimization, International Conference on Computer Systems and Technologies - *Compsystech'2005*.
[15] Zarni Aung and Win Zaw. "Permission-Based Android Malware Detection", International Journal of Scientific and Technology Research, Volume 2, Issue 3, 2013.
[16] L. Garfinkel and M. Rosenblum "When virtual is harder than real: Security challenges in virtual machine based computing environments. In 10th Workshop on Hot Topics in Operating Systems", 2005
[17] Fagbola, T., Olabiyis, S., & Adigun, A. (2012): Hybrid GA-SVM for Efficient Feature Selection in E-mail Classification. Computer & Intelligent System, ISSN 2222-28263, vol. 3, No. 3
[18] Liyang, W., Yongyi, Y., Nishikawa, R. M. & Yulei, J. (2005). A study on several machine-learning methods for classification of malignant and benign clustered microcalcifications. IEEE Transactions on Medical Imaging, 24(3), pp. 371–380.
[19] Andrew Webb (2010), Statistical Pattern Recognition. London: Oxford University Press.
[20] A.H. Mark, Lloyd A. Smith, "feature Subset Selection: A Correlation Based Filter Approach", 1997.
[21] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss: Andromaly: A behavioral malware detection framework for android devices. Journal of Intelligent Information Systems, 38(1), January 2011, pp. 1610-190.
[22] G. Dini, F. Martinelli, A. Saracino, D. Sgandurra: MADAM: A multi-level anomaly detector for android malware, 2012.
[23] S. Lin, K. Ying, S. Chen and Z. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines". Expert Systems with Applications. 35, 2008, PP. 1817–1824.
[24] H. Debar ,"Towards a taxonomy of intrusion detection systems", Computer Network, pp. 805-822, April 2009.
[25] Kruegel, Chris et al (2004): "*Intrusion Detection and Correlation: Challenges and Solutions"*, Springer.
[26] J. Kennedy, Some issues and practices for particle swarms, in: IEEE Swarm Intelligence, Honolulu, USA, 2001.