N-grams Based Supervised Machine Learning Model for Mobile Agent Platform Protection against Unknown Malicious Mobile Agents

Pallavi Bagga[1], Rahul Hans[2], Vipul Sharma[3] *

[1] Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab (India)
[2] Assistant Professor, Department of Computer Science and Engineering, DAV University, Jalandhar, Punjab (India)
[3] Assistant Professor, Department of Computer Science and Engineering, Jaypee University of Information Technology, Solan (India)

Received 8 February 2017 | Accepted 20 March 2017 | Published 30 March 2017

Abstract

From many past years, the detection of unknown malicious mobile agents before they invade the Mobile Agent Platform has been the subject of much challenging activity. The ever-growing threat of malicious agents calls for techniques for automated malicious agent detection. In this context, the machine learning (ML) methods are acknowledged more effective than the Signature-based and Behavior-based detection methods. Therefore, in this paper, the prime contribution has been made to detect the unknown malicious mobile agents based on n-gram features and supervised ML approach, which has not been done so far in the sphere of the Mobile Agents System (MAS) security. To carry out the study, the n-grams ranging from 3 to 9 are extracted from a dataset containing 40 malicious and 40 non-malicious mobile agents. Subsequently, the classification is performed using different classifiers. A nested 5-fold cross validation scheme is employed in order to avoid the biasing in the selection of optimal parameters of classifier. The observations of extensive experiments demonstrate that the work done in this paper is suitable for the task of unknown malicious mobile agent detection in a Mobile Agent Environment, and also adds the ML in the interest list of researchers dealing with MAS security.

Keywords

Classification, Malicious Mobile Agents, Nested Cross Validation, N-gram Feature Extraction.

DOI: 10.9781/ijimai.2017.03.013

I. Introduction

A collection of executable programs known as a Mobile agent (MA) migrates from one execution platform to another in a heterogeneous network to perform various tasks on the behalf of its user [1]. The employment of mobile agents introduce many benefits to the distributed computing including network load reduction, overcoming network latency, executing dynamically, asynchronously and autonomously [2]. In many respects, a mobile agent is analogous to a computer virus, since it travels from one computer to another and it utilizes computer resources or it creates clones of itself to achieve its goals. The major difference between both is the usefulness of mobile agent and its friendly behavior. However, the mobile agents while moving in the network, brings with them the fear of Trojan horses, viruses and other invasive means or entities [3]. This is because the attacks can be occurred when the mobile agent traverses in the communication channel and there may be some muggers earwigging the network either to gain some of the information carried by the agent or information stored in the agent platform (i.e. passive attack) or mutating that information for their own advantage (i.e. active attack) [4, 37]. In recent years, numerous researchers have done considerable studies in order to prevent malicious mobile agents causing any harm to Mobile Agent Platform (MAP).

* Corresponding author.

E-mail addresses: pallavibagga315@gmail.com (Pallavi Bagga), rahulhans@gmail.com (Rahul Hans), vipuls85@gmail.com (Vipul Sharma).

Wahbe et al. [11] proposed a Sandboxing technique, which offers an isolated environment (or a restricted area) for the execution of suspected mobile agents. This isolation prevents the mobile agent from accomplishing specific code exercises, for example local file system interaction, and accessing system properties. Noordende et al. [12] proposed a Mansion API where the agents execute in a protected environment like Sandboxing technique. Additionally, the agents are authenticated based on the trust level between agent owners as well as platform owners. Marikkannu et al. [13] suggested a Dual checkpoint mechanism involving two gates, inner and outer for the mobile agent verification consisting of Digital signatures as well as checksum ensuring the validity of a mobile agent. Alfalayleh et al. [14] recommended a Code Signing mechanism in which the sign of originator on code is checked by agent platform for verifying that it has not been modified. Lee et al. [15] proposed a technique in which the agent byte code compiles the proof carried by mobile agent with the platform's security policy. Upon receipt, the agent platform employs a proof checker for the purpose of checking and verifying the security proof of incoming agent byte code. Ordille [16] proposed the use of Path history that enables the platform either to run the agent or discard it; and to decide the trust level, privileges, resources and services that should be acknowledged to the agent if it is permitted to. Path History contains the identities of the current platform as well as the next platform in the itinerary. Cao et al. [17] proposed the use of agent path history information on the role activation and permission activation. The roles activated for an agent will be filtered by path patterns whereas the permissions for roles will be finely tuned by a set of host patches. Furthermore, Idrissi et al. [34, 36] proposed an authentication process based on Diffie–Hellman Key Exchange integrated with digital signature DSA to prevent the vulnerabilities arisen due to the unavailability of authentication, which makes it well resistant to the Man in the Middle attack; as well as another mobile agent platform security technique based on Elliptic Curve Cryptography (ECC) and dynamic role assignments using Role Based Access Control (RBAC) policy.

Venkatesan et al. [18] proposed Malicious Identification Police (MIP) that uses Attack Identification Scanner (AIS) to scan the incoming agent byte code in order to diagnose the maliciousness in it. In Policy Based MIP proposed by Venkatesan et al. [19], the privileges of an agent are also checked in addition to AIS [18], to know if it wants to do more than the privileges granted to it. Otherwise, Intelligent AIS (IAIS) decides to start the lexical analyzer by its own decision, where agent byte code is turned into tokens and diagnose the non-match tokens by comparing with tokens present in the Knowledge Base (KB). Afterwards, unknown tokens are executed and tested in an isolated environment to check for their malicious intentions and updates the KB containing malicious codes, with the newly diagnosed (if any) vicious code. In order to fend off this waiting time of the agent, Venkatesan et al. [20] further included the agent clones to handle multiple incoming agents simultaneously. Additionally, the pipelining concept was introduced by separating the operations i.e. the tasks of scanning, pattern extracting and detecting unknown codes are performed by different agents, which ultimately reduces the time complexity. Clearly, many researchers have been buckled down in the field of MAP security. However, the unknown malicious mobile agent detection before invading the MAP is still a challenge and a concern owing to the growth of malicious agents in recent years.

Nowadays, malicious code detection techniques employ one of these two approaches: Signature-based or Behavior-based. Signature-based methods involve the identification of distinctive tokens in the binary code [5]; whereas Behavior-based methods rely on the rules created by the experts that define the malicious behavior or non-malicious behavior of code [6]. While being very precise, signature-based methods are unable to diagnose previously unknown malicious codes whereas behavior-based methods can only detect the presence of malicious content after the code has been executed [7]. Realizing the necessity of a detection method for the unknown malicious code, in recent years, the machine learning algorithms or Classification Algorithms were magnificently employed which was highly inspired by the Text categorization problem [8]-[9], [23]-[25].

In this paper, an attempt has been made for detecting unknown malicious mobile agents using Machine Learning algorithms, which represents a novel contribution in the field of MAP security as per the survey done by the authors. This attempt addresses several facets of the detection challenge: mobile agent representation, classification and performance evaluation. The present work is also influenced by the objective to achieve very high classification accuracy rate while maintaining the low false negatives (i.e. misclassifying malicious agent as non-malicious). Though there are various representations of executable files: "Portable Executable (PE)", "Byte Sequence n-grams", and "plain-text string features" [9]; in this paper, n-gram representation of the agent executable is considered to be used as features for the classification process, since an extensive n-grams analysis is also one of the major focuses of this paper. N-grams are overlapping substrings obtained in a sliding window fashion [10].

The extracted n-gram features are then fed into four commonly used Classification algorithms: Naive Bayesian, SMO, IBK, J48 Decision Tree, for discriminating between two categories of agent classification (malicious mobile agent and non-malicious mobile agent), which is supported by WEKA tool [21]. The extensive experiments are performed on a collection of 80 files, in which half of the total files are malicious. The experimental results are evaluated based on standard performance evaluation measures such as "Sensitivity Rate", "Specificity Rate", "Positive Predictive Value", "Negative Predictive Value", "F-score", "Receiver Operating Characteristics – Area Under Curve", "Miss Rate", "Fall out" and "Accuracy Rate", while employing the 5-fold nested cross validation scheme.

II. Material and Methods

A. Dataset Used

To the best of author's knowledge, there is no standard data set available for the detection of malicious mobile agents. Therefore, the benchmark dataset of malicious files known as CSDMC20101[1] API sequence corpus, containing Windows API/System-Call trace files, is selected for the purpose of classification. The dataset contains 388 files involving 320 malware traces as well as 68 benign traces (considered as non-malicious in this paper). For the training dataset, only 40 malicious files and 40 non-malicious files are collected after random sampling (equal number for malicious and non-malicious files is considered in order to avoid the Class-imbalance problem). This standard dataset is preferable for the proposed approach since agent byte code can be viewed as a sequence of agent API function calls. This assumption is made on account of the previous studies of extracting API call sequences from byte codes [31],[32].

B. Performance Evaluation Measures

To evaluate the classification performance of detecting malicious mobile agents successfully, it is necessary to identify appropriate performance metrics. The measures derived from the Confusion Matrix (Figure 1) to calculate and be applied to classifier evaluation are described in Table I [26]. The confusion matrix indicates the correct and incorrect classification outcomes predicted by the classifier when compared with the actual classification outcome. The measures other than Accuracy Rate and Misclassification Rate are considered to figure out whether the present framework holds good for the classification of either malicious mobile agents or non-malicious mobile agents or both.

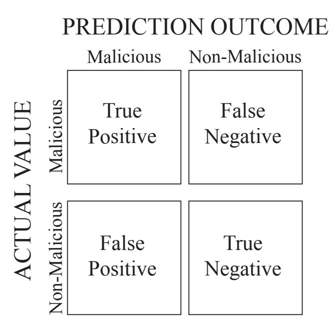

Fig. 1. Confusion Matrix to evaluate the performance of classifier.

- *True Positives* (TP): Number of malicious agents classified as malicious.
- *True Negatives* (TN): Number of non-malicious agents classified as non-malicious.
- *False Negatives* (FN): Number of malicious agents classified as non-malicious.

1 http://www.csmining.org/index.php/malicious-software-datasets-.html

TABLE I. Performance Evaluation Measures for Classification of Malicious Mobile Agents

Metric	Definition	Formula	Expected Value
Sensitivity	Also known as True Positive Rate (TPR) or Recall. It evaluates the ability of a classifier to correctly identify an agent as malicious.	$TP/(TP+FN)$	Maximum
Specificity	Also known as True Negative Rate (TNR). It evaluates the ability of a classifier to correctly identify an agent as non-malicious.	$TN/(TN+FP)$	Maximum
Positive Predictive Value (PPV)	Also known as Precision. It is the percentage of agents classified as malicious which are truly malicious.	$TP/(TP+FP)$	Maximum
Negative Predictive Value (NPV)	It is the percentage of agents classified as non-malicious which are truly non-malicious.	$TN/(TN+FN)$	Maximum
Miss Rate	Also known as False Negative Rate (FNR). It evaluates the proportion of malicious agents that are classified as non-malicious.	$FN/(TP+FN)$	Minimum
Fall out	Also known as False Positive Rate (FPR). It evaluates the proportion of non-malicious agents that are classified as malicious.	$FP/(FP+TN)$	Minimum
ROC-AUC	The curve is drawn by plotting the TPR against the FPR at different threshold settings.	NA	Between 0.9 and 1
Accuracy	It evaluates the ability of a classifier in classifying the whole dataset.	$(TP+TN)/(TP+FP+TN+FN)$	Maximum
F-measure	Also known as F-score. It is an evaluation of classifier's accuracy, which combines both the precision as well as the recall as a harmonic mean.	$2.Precision.Recall/(Precision+Recall)$	Maximum

Note: NA means Not Applicable

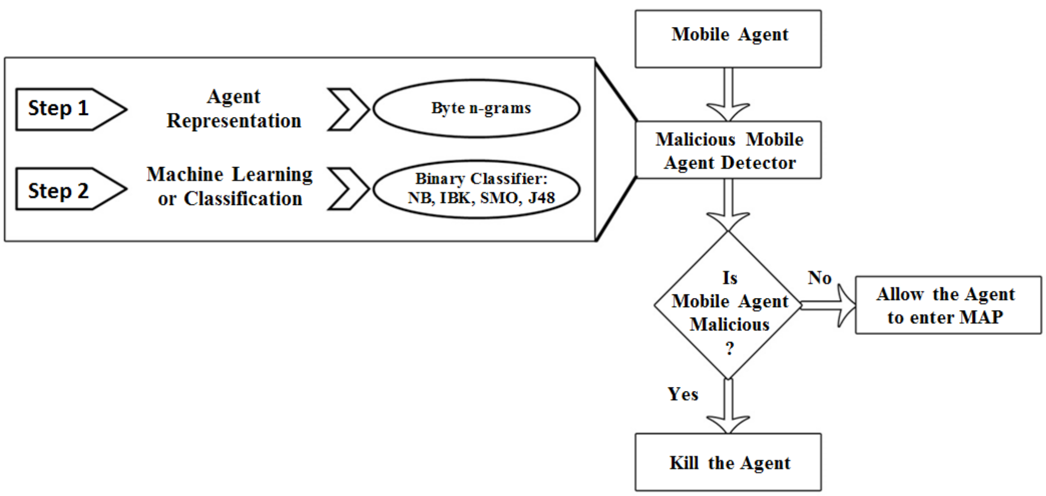

Fig. 2. Present Methodology for Malicious Mobile Agent Detection.

- *False Positives* (FP): Number of non-malicious agents classified as malicious.

C. Methodology

The major objectives of present methodology are as follows:

- To automate the detection of malicious mobile agents before they conquer the Mobile Agent Platform.
- To evaluate the performance of n-gram representation of mobile agent.
- To use Machine Learning algorithms for the task of unknown malicious mobile agent detection.
- To scrutinize the performance of various classifiers for classifying the mobile agents.

The methodology used in this paper is shown in Figure 2. It mainly consists of two consecutive steps: n-gram feature extraction of mobile agent and classification. These steps are described in detail in subsequent sub-sections.

1. Mobile Agent Representation using Byte n-grams – Data Preparation

A standard n-gram analysis is used to extract features from the malicious and non-malicious files. This method is purely machine-learning based method and exploits Natural Language Processing (NLP) also [30]. The n-grams are extracted in a sliding-window fashion, where a window of fixed length (n) slides one byte at a time. In general, n-grams are all substrings of a larger string with length "n" [24]. In present context, byte n-grams are viewed as API call based features. Many researches in recent years have released the importance of n-gram based methods in malware detection, since this technique of extracting features is simple and easy to implement. Each n-gram is analogous to a word or a term of a text document in the Text Categorization problem. For instance, there are eight 3-grams in the text "abc_dabc_e": "abc", "bc_", "c_d", "_da", "dab", "abc", "bc_" and "c_e". For the preparation of data, the unique n-grams are identified in all the mobile agent files and are merged together. In above example, there are only six distinct n-grams i.e. "abc", "bc_, "c_d", "_da", "dab", "c_e". The procedure of n-gram extraction repeats for different values of n. To limit the experiments for present study, the varying n-grams are employed with the value of n ranging from 3 to 9 only. This is because if the value of n increases, the number of unique n-gram features also increases. The number of distinct n-grams extracted from dataset files is 1403, 2236, 3074, 4055, 5137, 6445 and 7727 for 3-gram, 4-gram, 5-gram, 6-gram, 7-gram, 8-gram and 9-gram respectively.

2. Classification

Since the unknown mobile agent can be classified either malicious or non-malicious, the Binary Classification is taken into account. The standard commonly used classification algorithms such as Naïve Bayesian [22], Instance based Learner [22], Sequential Minimal Optimization [27]-[29], and J48 Decision Tree [22], are implemented. These classification algorithms differ in performance within different domains. In this paper, the best fitted algorithm for the dataset has been identified by the experimentation as shown in the subsequent section.

i) *IBK*

IBK is a WEKA implementation of k-Nearest Neighbor (k-NN). In general, the nearest neighbor classifiers compare a given test tuple with the identical training tuples. The training tuples are characterized using n features. Each tuple represents a point in an n-dimensional space. Hence, all the training tuples are exemplified in an n-dimensional feature space. When an unknown tuple is given as an input, a k-NN classifier explores the feature space for the closest k training tuples to the unknown tuple [22].

The closeness is defined in terms of Distance Metrics such as Chebyshev distance, Manhattan distance, and Euclidean distance. The unknown tuple is labeled with the most common class among its k-nearest neighbors. The value of k is usually an odd number to avoid tied votes; however, choosing the value of k is very analytical. The smaller value of k indicates the higher influence of noise on the result whereas the larger value of k makes the classification computationally very expensive. The pseudo code of IBK is shown in Algorithm 1.

Algorithm 1. IBK classification algorithm

Algorithm IBK(k, X, Y, x)
//Input: k- an integer odd value (number of nearest neighbors), X- Training data consisting of n tuples, Y- Class Labels of X, x- Unknown tuple
//Output: Class label of x
1. **for** i ← 1 to n do
2. compute distance (X_i, x)
3. **end for**
4. sort the distances in ascending order
5. select the first k points from the sorted list (these are the k nearest training tuples to unknown tuple)
6. **return** class label that belongs to the majority of k selected tuples

ii) *Naïve Bayesian*

This classifier is so called because it relies on the Bayesian Theorem. Moreover, it is called "Naive", because it assumes the independence between every pair of attributes (or features), which is known as "class conditional independence" [22]. The classifier takes an unknown tuple as an input, for which the class label is not known, and returns a class label as an output for which the maximum probability is obtained as per the probabilistic calculations. This classifier is particular suited for the higher dimensionality of features. The pseudo code of Naïve Bayesian is shown in Algorithm 2.

Algorithm 2. Naïve Bayesian classification algorithm

Algorithm naiveBayesian(D,X,m,n,A)
//Input: D - Training set of tuples and their associated class labels, n- number of attributes, A - set of attributes (A_1,A_2,….A_n), X - n-dimensional attribute vector (x_1,x_2,….x_n), m – number of classes
//Output: Class of tuple X
1. **for** i ← 1 to m do
2. calculate the Posterior probability conditioned on X i.e. $P(C_i|X)$ using
Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (1)$$

where in Equation (1),
P(X) is constant for all classes,
$P(C_i) = \frac{|C_{i,D}|}{|D|}$, where $|C_{i,D}|$ is the number of training tuples of Class C_i in D

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i) \quad (2)$$

and,in Equation (2), x_k refers to the value of attribute A_k for tuple X
3. **end for**
4. **return** class label with maximum $P(C_i|X)$

iii) *J48 Decision Tree*

A Decision tree classifies the tuples as per a set of tree-structured if-then-rules. Each internal node represents a test on an attribute (or a feature), each branch represents the outcome of test, whereas each leaf node holds a class label. J48 is a WEKA implementation of the C4.5 decision tree. It consists of two steps: the decision tree induction and the tree pruning [22]. The pseudo code of decision tree algorithm is shown in Algorithm 3.

Algorithm 3. J48 Decision Tree classification algorithm

//Input: D- set of training tuples and their associated class labels, A- attribute list
//Output: Decision Tree
1. create a node N
2. **if** all tuples of D belong to same class C, **then**
3. **return** N as a leaf node labeled with class C and terminate.
4. **if** A is empty, **then**
5. **return** N as a leaf node labeled with the most common class in D
(Majority Voting)
6. apply Gain Ratio feature selection method to find the best criterion 'a' ∈ A
7. label N with 'a'
8. **for** each value j of 'a' **do**
9. grow a branch from N with condition a=j
10. let Dj be the set of tuples in D with a=j
11. **if** Dj is empty **then**
12. add a leaf node labeled with most common class in D to N
13. **else** add the node returned by decisionTree(Dj, A-a) to N
14. **end for**
15. **return** N

iv) *SMO*

SMO is a WEKA implementation of Support Vector Machine (SVM). SVM was originated from statistical learning theory with the objective to find the solution of interested problem without solving a more difficult problem as an intermediate step [27]. The statistical learning theory offers a framework that helps to choose the hyper plane space such that it diligently symbolizes the underlying function in the target space [28].

SVM avoids over-fitting to the training dataset.

Given a training set of instance-label pairs (x_i, y_i), i=1,….,N where $x_i \in R^n$ and $y \in \{1,-1\}^l$, SVM finds the solution of an optimization problem as shown in Algorithm 4 [29]. The training tuples x_i are represented into a higher (possibly infinite) dimensional space using function $\emptyset$. In other words, the kernel $\emptyset$ is used to transform data from input to the feature space. There are four basic kernels: Linear, Polynomial, Radial Basis Function (RBF) and Sigmoidal.

Algorithm 4. SMO Optimization Problem

Optimization Problem:
Given Constraints in Equation (3):

$$y_i(w^T \emptyset (x_i) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad (3)$$

Minimize the Error Function as mentioned in Equation (4):

$$\frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \quad (4)$$

where,
C>0 is the penalty parameter of the error term, and is called Capacity Constant. The choice of C is made carefully in order to avoid over-fitting. Larger the C, more the error is penalized.
w is the vector of coefficients.
b is a constant.
ξ_i represents parameters for handling non-separable data (inputs)
i labels the N training cases
x_i represents the independent variables.

III. Results and Discussions

The classification of mobile agent into two categories based on their n-gram features has been performed on 80 agent files of dataset of API calls sequence. An extensive setting of parameters is done to optimize the performance of each classification algorithm (NB, SMO, IBK and J48), such as "value of k", "distance measure", or "nearest neighbor search algorithm" in IBK, "pruning", or "confidence factor" in J48 decision tree, "complexity parameter", or "kernel" in SMO. The nested five-fold cross validation scheme is performed to obtain unbiased evaluation results [33]. In nested five-fold cross validation method, the data is randomly divided into five disjoint folds. The four folds are used for tuning of classifier parameters (using cross validation scheme) and then the tuned classifier is validated on left out fold.

This procedure repeats for five times, each time with different left-out folds. This nesting of cross validation loops avoids so-called resubstitution-bias [33]. Additionally, the standard parameters such as Sensitivity, Specificity, ROC, PPV, NPV, FNR, FPR, F-measure and Accuracy, evaluate the performance results and the results of all iterations are averaged to get the final outcome. It has been evidenced that the performance of present work highly depends on the choice of classifier.

The results of various classifiers (NB, SMO, IBK and J48) for different values of n for n-grams i.e. n=3 to 9 have been investigated and are presented in Table II. To help with nested cross validation, WEKA tool has been used to adjust the classifier settings repeatedly in order to get the results on suitable parameter values. The results demonstrate that IBK classifier, J48 classifier, SMO classifier and NB classifier gives maximum accuracy rate of 96.25%, 97.50%, 95.00% and 93.75% respectively (as shown in Figure 3), while maintaining the miss rate of 2.50%, 2.50%, 5.00% and 7.50% for trigrams, 9-grams, 5-grams to 9-grams and 7-grams respectively (as depicted in Figure 4) in distinguishing malicious files from non-malicious. The value for Area under ROC curve is more than 0.92 for each classifier. IBK gives maximum sensitivity of 97.50 % and specificity of 95.00% for 3-gram features. J48 gives maximum sensitivity of 97.50 % as well as specificity of 97.50% for 9-gram features as shown in Figure 5 and Figure 6. Moreover, the highest values of PPV and NPV (97.50% each) belong to J48 classifier for 9-grams.

TABLE II. Results of Different Classifiers for Different Values of N in N-Gram

n	Classifier	Accuracy Rate (%)	Miss Rate (%)	Fall out (%)	PPV (%)	NPV (%)	F-measure (%)	ROC area
3	Bayesian	85.00	15.00	15.00	85.00	85.00	85.00	0.94
	SMO	92.50	7.50	7.50	92.50	92.50	92.50	0.93
	J48	95.00	2.50	7.50	92.86	97.37	95.12	0.94
	IBK	96.25	2.50	5.00	95.12	97.44	96.30	0.97
4	Bayesian	87.50	12.50	12.50	87.50	87.50	87.50	0.93
	SMO	93.75	5.00	7.50	92.68	94.87	93.83	0.94
	J48	93.75	5.00	7.50	92.68	94.87	93.83	0.94
	IBK	93.75	5.00	7.50	92.68	94.87	93.83	0.94
5	Bayesian	90.00	12.50	7.50	92.11	88.10	89.74	0.94
	SMO	95.00	5.00	5.00	95.00	95.00	95.00	0.95
	J48	95.00	2.50	7.50	92.86	97.37	95.12	0.94
	IBK	93.75	5.00	7.50	92.68	94.87	93.83	0.93
6	Bayesian	91.25	10.00	7.50	92.31	90.24	91.14	0.95
	SMO	95.00	5.00	5.00	95.00	95.00	95.00	0.95
	J48	96.25	2.50	5.00	95.12	97.44	96.30	0.95
	IBK	92.50	2.50	12.50	88.64	97.22	92.86	0.93
7	Bayesian	93.75	7.50	5.00	94.87	92.68	93.67	0.96
	SMO	95.00	5.00	5.00	95.00	95.00	95.00	0.95
	J48	95.00	5.00	5.00	95.00	95.00	95.00	0.94
	IBK	92.50	2.50	12.50	88.64	97.22	92.86	0.93
8	Bayesian	92.50	7.50	7.50	92.50	92.50	92.50	0.96
	SMO	95.00	5.00	5.00	95.00	95.00	95.00	0.95
	J48	96.25	2.50	5.00	95.12	97.44	96.30	0.96
	IBK	92.50	2.50	12.50	88.64	97.22	92.86	0.94
9	Bayesian	92.50	10.00	5.00	94.74	90.48	92.31	0.97
	SMO	95.00	5.00	5.00	95.00	95.00	95.00	0.95
	J48	97.50	2.50	2.50	97.50	97.50	97.50	0.97
	IBK	92.50	5.00	10.00	90.48	94.74	92.68	0.93

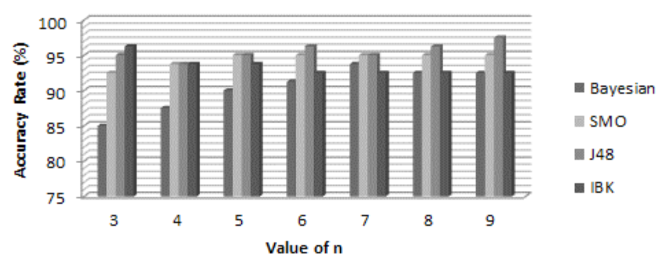

Fig. 3 Graph showing Accuracy Rate of Different Classification Algorithms.

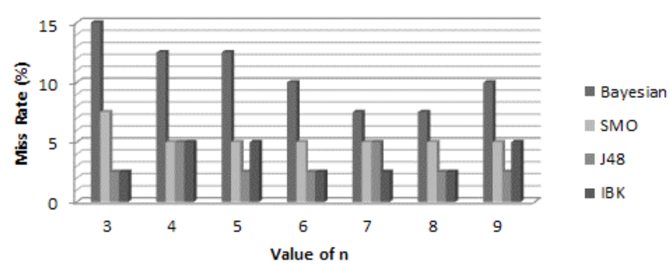

Fig. 4 Graph showing Miss Rate of Different Classification Algorithms.

This means IBK and J48 decision tree are the best in distinguishing actual malicious agents (positives) and actual non-malicious agents (negatives) using 3-gram and 9-gram features respectively. The performance of other classification algorithms reduces with the increase in number of features.

It is shown in Figure 5 that the sensitivity rate increases up to n=7 using NB classifier and then decreases. Using SMO, the sensitivity rate increases up to n=4 and then remains the same with further increase in value of n. Using J48 and IBK classifiers, sensitivity rate is minimum for n=3, which decreases and then again increases with increase in value of n. The Figure 6 indicates that the minimum specificity rate (85.00%) is obtained using NB classifier for 3-grams, which further increases with the increase in value of n. SMO and J48 classifiers provide the minimum specificity rate of 92.50%. From 5-gram to 9-grams, specificity is constant with SMO classifier whereas from 6 to 8-grams, IBK increases with increase in value of n.

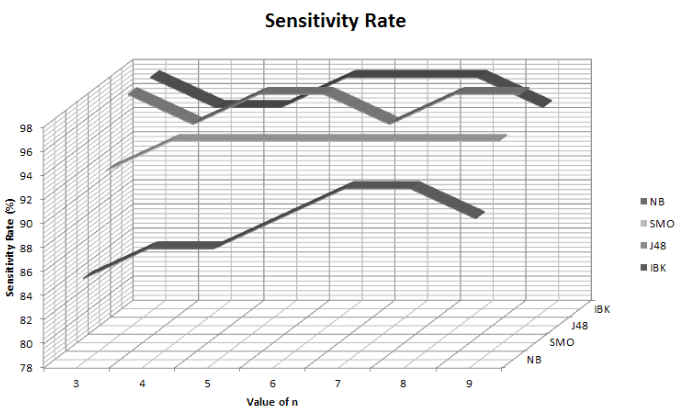

Fig. 5 Graph showing Sensitivity Rate of Different Classification Algorithms.

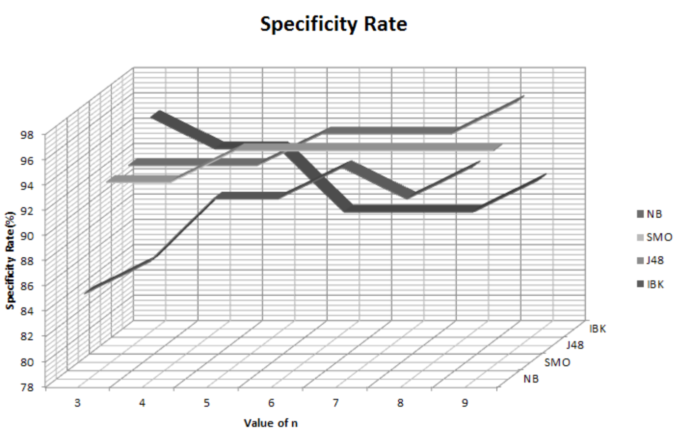

Fig. 6 Graph showing Specificity Rate of Different Classification Algorithms.

IV. Conclusions and Future Scope

This paper aims for probing the suitability of machine learning algorithms for the task of classification of mobile agent either malicious or non-malicious in a Mobile Agent Environment using a specific dataset. In particular, n-grams are used as features during the classification process. Different classification algorithms used are Naïve Bayesian, Sequential Minimization Optimization, Instance Based Learner, and J48 Decision Tree. J48 decision tree algorithm gives higher accuracy rate of 97.50% and miss rate of 2.50% for 9-grams whereas IBK gives higher accuracy rate of 96.25% and miss rate of 2.50% for 3-grams, as compared to other classification algorithms. In other words, out of all the classifiers, J48 gives better result with 7727 features whereas IBK with 1403 features, but J48 increases the computational cost due to large feature space. Therefore, IBK is declared to be the best classifier. Clearly, the optimistic results boost the use of present research for MAP protection.

In near future, the work can be extended with the use of more different classifiers with higher values of n for n-gram features in order to evaluate the performance of classification task. Moreover, large number of n-gram features burdens the classification process; therefore, feature selection methods can be applied in future. The work can even be done on different datasets, since the classifiers may give different results on different datasets.

References

[1] A. Aneiba, and S.J. Rees, “Mobile Agents Technology and Mobility”, in *Proceedings of the 5th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting,* 2004. pp. 14-20.

[2] D.B. Lange, and M. Oshima, “Seven good reasons for Mobile Agents”, *Communications of the ACM*, vol. 42, no. 3, pp. 88-89, 1999.

[3] L.L. Thomsen, and B. Thomsen, “Mobile Agents – The new paradigm in computing”, *ICL- The Systems Journal*, vol. 12, pp. 14-40, 1997.

[4] R. Oppliger, “Security issues related to mobile code and agent-based systems”, *Computer Communications*, vol. 22, no. 12, pp. 1165-1170, 1999.

[5] D. Venugopal, and G. Hu, “Efficient signature based malware detection on mobile devices”, *Mobile Information Systems*, vol. 4, no. 1, pp. 33-49, 2008.

[6] W. Ma, P. Duan, S. Liu, G. Gu, J. Liu, “Shadow attacks: automatically evading system-call-behavior based malware detection”, *Journal in Computer Virology*, vol. 8, no. 1, pp. 1-13, 2012.

[7] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, D. Shlomi, and Y. Elovici, “Unknown Malcode Detection Using OPCODE Representation”, *Intelligence and Security Informatics*, vol. 5376, pp. 204-15, 2008.

[8] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, “Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey”, *Information Security Technical Report*, vol. 14, no. 1, pp. 16-29, 2009.

[9] J.Z. Kotler, and M.A. Maloof, “Learning to detect malicious executables in the wild”, in *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Aug 22, New York, ACM Press, pp. 470-478, 2004.

[10] S. Jain, and Y.K. Meena, “Byte Level n-Gram Analysis for Malware Detection”, in *Proceedings of the 5th International Conference on Information Processing*, ICIP 2011, Aug 5-7, Banglore, India, Springer Berlin Heidelberg, pp. 51-59, 2011.

[11] R. Wahbe, S. Lucco, T.E. Anderson, and S.L. Graham, “Efficient Software-Based Fault Isolation”, in *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles*, SOSP ’93, Dec 5-8, Asheville, North Carolina, USA, ACM Press, pp. 203-216, 1993.

[12] G. Noordende, F. M. Brazier, and A.S. Tannenbaum, “A security framework for a mobile agent system”, in *Proceedings of the 2nd International Workshop on Security in Mobile Multiagent Systems*, SEMAS 2002, Bologna, Italy, pp. 43-50, 2002.

[13] P. Marikkannu, and A. Jovin, “A Secure Mobile Agent System against Tailgating Attacks”, *Journal of Computer Science*, vol. 7, no. 4, pp. 488-492, 2011.

[14] M. Alfalayeh, and L. Brankovic L, “An overview of security issues and techniques in mobile agents”, in *Proceedings of the 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security,* pp. 59-78, 2005.

[15] P. Lee, and G. Necula, “Research on proof-carrying code for mobile-code security”, in *DARPA workshop on foundations for secure mobile code,* pp. 26-28, 1997.

[16] J.J. Ordille, “When agents roam, who can you trust?”, in *Proceedings of the IEEE First Annual conference on Emerging Technologies and Applications in Communications,* pp. 188-191, 1996.

[17] C. Cao, and J. Lu, “Path-history-based access control for mobile agents,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 21, no. 3, pp. 215-225, 2006.

[18] S. Venkatesan, and S. Chellappan, “Protection of mobile agent platform through Attack Identification Scanner (AIS) by Malicious Identification Police (MIP)”, in *Proceedings of the IEEE First International Conference on Emerging Trends in Engineering and Technology (ICETET ’08)*, IEEE, pp. 1228-1231, 2008.

[19] S. Venkatesan, C. Chellappan, T. Vengattaraman, P. Dhavachelvan, and Vaish, “Advanced mobile agent security models for code integrity

and malicious availability check", *Journal of Network and Computer Applications*, vol. 33, no. 6, pp. 661-671, 2010.
[20] S. Venkatesan, R. Baskaran, C. Chellappan, A. Vaish, and P. Dhavachelvan, "Artificial immune system based mobile agent platform protection," *Computer Standards & Interfaces*, vol. 35, no. 4, pp. 365-373, 2013.
[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reuteman, and I.H. Witten, "The WEKA data mining software: an update", *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10-18, 2009.
[22] J. Han, "Data mining: concepts and Techniques", 2000.
[23] D. Gavriluţ, M. Cimpeşu, D. Anton, and L. Ciortuz, "Malware Detection Using Machine Learning", in *Proceedings of the International Multiconference on Computer Science and Information Technology,* Oct 12, IEEE, pp. 735-741, 2009.
[24] T. Abou-Assaleh, N. Cercone, V. Kešelj, and R. Sweidan, "N-gram-based detection of new malicious code", in *Proceedings of the IEEE 28th Annual International Conference on Computer Software and Applications*, pp. 41-42, 2004.
[25] I. Santos, Y.K. Penya, J. Devesa, and P.G. Bringas. "N-grams-based File Signatures for Malware Detection", in *Proceedings of the 2009 International Conference on Enterprise Information Systems*, pp. 317-320, 2009.
[26] M. Sokolova, and G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Information Processing & Management*, vol. 45, no. 4, pp. 427-437, 2009.
[27] R. Burbidge, B. Buxton, "An Introduction to Support Vector Machines for Data Mining", Keynote Speakers, young OR12, pp. 3-15, 2001.
[28] V. Jakkula, "Tutorial on Support Vector Machine (SVM)", School of EECS, Washington State University, 2006.
[29] C.W. Hsu, C.C. Chang, C.J. Lin, "A Practical Guide to Support Vector Classification", *Technical Report*, Department of computer Science, National Taiwan University, 2003.
[30] C.Y. Suen, "n-Gram Statistics for Natural Language Understanding and Text Processing", *IEEE Transactions on Pattern Analysis*, vol. PAMI-1, no. 2, pp.164-172, 1979.
[31] T.T. Nguyen, H.V. Pham, P.M. Vu, and T. T. Nguyen, "Learning API Usages from Bytecode: A Statistical Approach", in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16),* May 14-22, Austin, Texas, ACM Press, pp. 416-427, 2016.
[32] Y. Qiao, Y. Yang, J. He, C. Tang, and Z. Liu, "CBM: Free, Automatic Malware Analysis Framework Using API Call Sequences", in *Proceedings of the Seventh International Conference on Intelligent Systems and Knowledge Engineering,* Dec 2012, Beijing, China, Springer Berlin Heidelberg, pp. 225-236.
[33] S. Varma, and R. Simon, "Bias in error estimation when using cross validation for model selection", *BMC Informatics*, vol. 7, no. 1, 2006.
[34] H. Idrissi, E. M. Souidi, and A. Revel, "Security of mobile agent platforms using access control and cryptography", in *Proceedings of the 9th KES international conference (KES-AMSTA)*, pp. 27–39, 2015.
[35] H. Idrissi, A. Revel, and E. M. Souidi, "Security of Mobile Agent Platforms using RBAC based on Dynamic Role Assignment", *International Journal of Security and Its Applications*, vol. 10, no. 4, pp.117-134, 2016.
[36] M. Malathy, S.J. Smilee, J. N. Samuel, "Secure Mobile Agent in M-Commerce over Internet", in *Proceedings of International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), pp. 1-5, 2016.*

Pallavi Bagga

Pallavi Bagga is presently working as an Assistant Professor in the Department of Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab, India. She has accomplished her Master of Technology in Computer Science & Engineering from DAV University, Jalandhar, Punjab, India in 2016. Earlier, she has completed her Bachelor of Technology in Computer Science & Engineering from Punjab Technical University, Punjab, India in 2014. Her predominant areas of research interests include Distributed Computing and Machine Learning. Nowadays, she is actively pursuing her research career in the sphere of Mobile Agents System Security using Machine Learning Algorithms.

Rahul Hans

Rahul Hans is currently working as an Assistant Professor in the Department of Computer Science and Engineering at DAV University, Jalandhar, Punjab, India. He has received his B.Tech degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, Punjab, India and subsequently his M.Tech degree in Computer Science and Engineering from Guru Nanak Dev University, Amritsar, Punjab, India. He has a teaching experience of more than four years. He is also a member of Computer Society of India (CSI). He has published 16 research papers in various International Conferences and Journals. His research areas include Security and Fault tolerance in Mobile Agent Systems in Distributed Computing as well as the Machine Learning.

Vipul Sharma

Dr. Vipul Sharma is currently working as an Assistant Professor (Senior Grade) in the Department of Computer Science and Engineering at Jaypee University of Information Technology, Solan, India. He has received his B.Tech. Degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, Punjab, India and subsequently his M.E. and Ph.D. Degrees in Computer Science & Engineering from UIET, Panjab University, Chandigarh, India. His Ph.D. thesis focused on use of machine learning techniques to associate low level image features with the query concepts, relevance feedback techniques to learn users' intention and feature selection to find an optimal set of features that can not only capture the texture of images but also enhances the semantic interpretability for reducing the semantic gap in medical image retrieval. He is actively pursuing research in Medical Image Processing. He is the awardee of TEQIP-II fellowship during Ph.D. He also received the UIET, Panjab University Research Award-2015 and a cash prize of Rs 10000 for his outstanding publication in the Medical & Biological Engineering & Computing Journal (SCI Journal). He is a member of many professional bodies including Society of Photo-Optical Instrumentation Engineers, USA; Engineering in Medicine & Biology Society; Association for Computing Machinery; IEEE; IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence; Multimedia Computing; IEEE Young Professionals. He has published 16 research articles in reputed journals and conferences. His current research includes development of effective classification and retrieval techniques for the retrieval of medical images from large image databases.