Anomaly Network Intrusion Detection Based on Improved Self Adaptive Bayesian Algorithm

Dewan Md. Farid
Dept. of CSE, Jahangirnagar University, Dhaka-1342, Bangladesh
Email: dmfarid@uiu.ac.bd

Mohammad Zahidur Rahman
Dept. of CSE, Jahangirnagar University, Dhaka-1342, Bangladesh
Email: rmzahid@juniv.edu

Abstract—**Recently, research on intrusion detection in computer systems has received much attention to the computational intelligence society. Many intelligence learning algorithms applied to the huge volume of complex and dynamic dataset for the construction of efficient intrusion detection systems (IDSs). Despite of many advances that have been achieved in existing IDSs, there are still some difficulties, such as correct classification of large intrusion detection dataset, unbalanced detection accuracy in the high speed network traffic, and reduce false positives. This paper presents a new approach to the alert classification to reduce false positives in intrusion detection using improved self adaptive Bayesian algorithm (ISABA). The proposed approach applied to the security domain of anomaly based network intrusion detection, which correctly classifies different types of attacks of KDD99 benchmark dataset with high classification rates in short response time and reduce false positives using limited computational resources.**

Index Terms—**anomaly detection, network intrusion detection, alert classification, Bayesian algorithm, detection rate, false positives**

I. Introduction

Network intrusion detection is the problem of detecting unauthorized use of computer systems over a network, such as the Internet. IDSs were introduced by James P. Anderson in 1980, an example of an audit trails would be a log of user access [1]. IDSs have become an integral part of today's information security infrastructures. In order to detect intrusion activities, many machine learning (ML) algorithms, such as Neural Network [2], Support Vector Machine [3], Genetic Algorithm [4], Fuzzy Logic [5], and Data Mining [6], etc have been widely used to the huge volume of complex and dynamic dataset to detect known and unknown intrusions. It is very important for IDSs to generate rules to distinguish normal behaviors from abnormal behavior by observing dataset, which is the record of activities generated by the operating system that are logged to a file in chronologically sorted order. IDSs using ML algorithms aim to solve the problems of analyzing huge volumes of dataset and realizing performance optimization of detection rules. IDSs detect attacks on computer systems and signal an alert to the Computer Emergency Response Team (CERT). The network intrusion detection area is the arrival of new, previously unseen attacks, because hackers are very inventive and they use newer ways to disrupt the normal operation of servers and users. Anomaly detection detects new attacks which has not presented in the dataset by observing system activities and classifying it as either normal or anomalous. An important research challenge today is to develop adaptive IDSs to improve classification rates, and reduce false positives.

The Bayesian algorithm (BA) provides a probabilistic approach for classification [7], [8], which provides an optimal way to predict the class of an unknown example. It is widely used in many fields of data mining, image processing, bio-informatics, and information retrieval etc. It calculates conditional probabilities from a given dataset and used these conditional probabilities to find out the probabilities of belongingness to different classes. The unseen example is then categorized to that class, which assumes the maximum value. In this paper, based on a comprehensive analysis for the current research challenges we propose a new algorithm to address the problem of classification rates and false positives using improved self adaptive Bayesian Algorithm. This algorithm correctly classifies different types of attacks of KDD99 dataset with high detection accuracy in short response time, and also maximizes the detection rate (DR) and minimizes the false positives (FP). In our experiments proposed algorithm reduced the number of false positives by up to 90% with acceptable misclassification rates.

The remainder of this paper is organized as follows. Section II provides a review of IDSs and section III provides IDSs using machine learning algorithms. Section IV presents our proposed new algorithm. Experimental results are presented in section V. Finally, section VI makes some concluding remarks along with suggestions for further improvement.

II. Review of Intrusion Detection Systems

A. History of IDSs

In 1980, the concept of IDSs began with Anderson's seminal paper, introduced the notion that audit trails contained vital information that could be valuable in tracking misuse and understanding user behavior. His work was the start of host-based IDSs (HBIDSs). In 1986, Dr. Dorothy Denning published a model which revealed the necessary information for commercial IDSs development [9]. In 1988, multics intrusion detection and alerting system (MIDAS), an expert system using P-BEST and LISP was developed [10]. Haystack was also developed in this year using statistics to reduce audit trails [11]. In 1989, wisdom & sense (W&S) was a statistics-based anomaly detector developed, that created rules based on statistical analysis, and then used those rules for anomaly detection [12]. In 1990, Heberlein first introduced the idea of network IDSs, development of Network Security Monitor (NSM), and hybrid IDSs [13] and Lunt proposed SRI named intrusion detection expert system (IDES), a dual approach of a rule-based expert system and a statistical anomaly detection, which ran on Sun Workstations and could consider both user and network level data [14]. Also in the early 1990s the commercial development of IDSs were started and the Time-based inductive machine (TIM) did anomaly detection using inductive learning of sequential user patterns in Common LISP on a VAX 3500 computer [15]. In 1991, distributed IDSs (DIDS), an expert system created by the researchers of University of California [16], and the network anomaly detection and intrusion reporter (NADIR) a statistic based anomaly detector and also an expert system developed by Los Alamos National Laboratory's Integrated Computing Network (ICN) [17]. In 1993, Lunt proposed the Next-generation Intrusion Detection Expert System by developing SRI followed IDES using artificial neural network [18]. The Lawrence Berkeley National Laboratory introduced rule language called Bro for packet analysis from libpcap dataset in 1998 [19]. The audit data analysis and mining IDSs used tcpdump to build profiles of rules for classifications in 2001 [20].

B. *Types of IDSs*

Intrusion is a set of actions that attempt to compromise the confidentiality, integrity or availability of computer resources. IDSs collect information from a variety of systems and analyze the information for signs of intrusions. In general, IDS categorize into five types, which are described below:

1) Network IDSs (NIDSs) responsible for detecting attacks related to the network [21], [22]. NIDSs investigate incoming and outgoing network traffic by connecting with network devices to find suspicious patterns. If a NIDS has no additional information about the protected host, the malicious attacker can easily avoid detection by taking advantage of different handling by overlapping IP/TCP fragments by IDS and a target host [23].

2) Host-based IDSs (HBIDSs) usually are located in servers to examine the internal interfaces [24]. HBIDSs can either use standard auditing tools [25], or specially instrumented operating system [26], or application platforms [27]. It detects intrusions by analyzing system calls, application logs, file-system modifications, and other host activities related to the machine.

3) Protocol-based IDSs (PIDSs) monitor the dynamic behavior and state of the protocol used the web server. PIDSs sit at the front end of a web server, monitoring and analyzing the HTTP protocol stream. It understands the HTTP protocol to protect web server by filtering IP address or port number.

4) Application protocol-based IDSs (APIDSs) monitor and analysis on a specific application protocol or protocols between a process, and group of servers that is used by the computer system. APIDSs can be sitting between a web server and the database management system that monitoring the SQL protocol specific to the business logic. Generally, APIDSs look for the correct use of the protocol.

5) Hybrid IDSs (HIDSs) combines two or more intrusion detection approaches. HIDS provide alert notification from both network and host-based intrusion detection devices.

C. *Detection Models of IDSs*

Detection rate (DR) is defined as the number of intrusion instances detected by the system divided by the total number of intrusion instances present in the dataset, and false positives (FP) is an alarm, which rose for something that is not really an attack. There are two types of detection models for IDSs, which are described below:

1) Misuse or signature-based IDSs are also known as pattern-based IDSs. It performs simple pattern matching to match a pattern corresponding to a known attack type in the database. DR of these IDSs is relatively low, because attacker will try to modify the basic attack signature in such a way that will not match the known signatures of that attack and it cannot detect a new attack for which a signature is not yet installed in the database.

2) Anomaly-based IDSs tries to identify new attacks by analyzing strange behavior from normal behaviors. It has a relatively high detection rate for new types of intrusion. The disadvantage is that in many cases there is no signal "normal profile" and anomaly-based systems tend to produce many false positives.

D. *Functions of IDSs*

IDSs are automated systems detecting and alarming of any situation where an intrusion has taken or is about to take place. According to the Common Intrusion Detection Framework (CIDF), generally IDSs consists of four components, like: sensors, analyzers, database, and response units. Most modern IDSs use multiple intrusion sensors, which obtain alerts from the large computational environment to maximize their trustworthiness. Analyzers use the input of the sensors, analyze the information gathered by these sensors, and return a synthesis or summary of the input. Today, machine learning algorithms have become an indispensable tool in the analyzers of IDSs. Database stores all alerts and support the analysis process. The response units carry out prescriptions controlled by the analyzers. The functions of IDSs are follows as [28]:

- Monitoring user's activity.

- Monitoring systems activity.
- Auditing system configuration.
- Assessing the data files.
- Recognizing known attack.
- Identifying abnormal activity.
- Managing audit data.
- Highlighting normal activity.
- Correcting system configuration errors.
- Stores information about intruders.

III. IDS USING MACHINE LEARNING

A. Bayes Rule

The Bayes rule provides a way to calculate the probability of a hypothesis based on its prior probability [7],[8]. The best hypothesis is the most probable hypothesis, given the observed data D plus any initial knowledge about the prior probabilities of the various hypotheses h (h is a hypothesis space containing possible target function). Bayes rule is defined in equation "(1)".

$$\textit{Bayes Rule: } P(h|D)=\frac{P(D|h)P(h)}{P(D)} \quad (1)$$

Here $P(h|D)$ is called the posterior probability, while $P(h)$ is the prior probability associated with hypothesis h. $P(D)$ is the probability of the occurrence of data D and $P(D|h)$ is the conditional probability. In many learning scenarios, the learner considers some set of candidate hypothesis, H and is interested in finding the most probable hypothesis h € H given the data D. Any such maximally probable hypothesis is called maximum posterior (MAP) hypothesis. The MAP hypothesis use Bayes rule to calculate the posterior probability of each candidate hypothesis. More exactly, h_{MAP} is a MAP hypothesis provided:

$$\begin{aligned} h_{MAP} &\equiv \operatorname{argmax}_{h\in H} P(h|D) \\ &= \operatorname{argmax}_{h\in H} \frac{P(h|D)P(h)}{P(D)} \\ &= \operatorname{argmax}_{h\in H} P(D|h)P(h) \end{aligned} \quad (2)$$

Finally, dropped the term $P(D)$ because it is a constant dependent of h. $P(D|h)$ is also called likelihood of the data D given h any hypothesis that maximizes $P(D|h)$ is called a Maximum Likelihood (ML) hypothesis, h_{ML}:

$$h_{ML} \equiv \operatorname{argmax}_{h\in H} P(D|h) \quad (3)$$

B. Naïve Bayesian Classifier

Naïve Bayesian (NB) classifier is a simple probabilistic classifier based on probability models that incorporate strong independence assumptions which often have no bearing in reality. The probability model can be derived using Bayes rule. Depending on the precise nature of the probability model, NB classifier can be trained very efficiently in a supervised learning. In many practical applications, parameter estimation for naïve Bayesian models uses the method of maximum likelihood. In spite of naïve design and apparently over-simplified assumptions, naïve Bayesian classifiers often work much better in many complex real-world situations. The NB classifier is given as input a set of training examples each of which is described by attributes A_1 through A_k and an associated class, C. The objective is to classify an unseen example whose class value is unknown but values for attributes A_1 through A_k are known and they are $a_1, a_2, \ldots, a_k$ respectively. The optimal prediction of the unseen example is the class value c such that $P(C=c_i|A_1=a_1,\ldots A_k=a_k)$ is maximum. By Bayes rule this probability equals to

$$\operatorname{argmax}_{c_i\in C} \frac{P(A_1=a_1,..A_k=a_k\,|\,C=c_i)}{P(A_1=a_1,...A_k=a_k)} P(C=c_i) \quad (4)$$

Where $P(C=c_i)$ is the prior probability of class c_i, $P(A_1=a_1,\ldots A_k=a_k)$ is the probability of occurrence of the description of a particular example, and $P(A_1=a_1,\ldots A_k=a_k|C=c_i)$ is the class conditional probability of the description of a particular example c_i of class C. The prior probability of a class can be estimated from training data. The probability of occurrence of the description of particular examples is irrelevant for decision making since it is the same for each class value c. Learning is therefore reduced to the problem of estimating the class conditional probability of all possible description of examples from training data. The class conditional probability can be written in expanded from as follows:

$$\begin{aligned} &P(A_1=a_1,\ldots A_k=a_k|C=c_i) \\ &\quad = P(A_1=a_1|\ A_2=a_2 \wedge\ldots A_k=a_k \wedge C=c_i) \\ &\quad\quad * P(A_2=a_2|\ A_3=a_3 \wedge\ldots A_k=a_k \wedge C=c_i) \\ &\quad\quad * P(A_3=a_3|\ A_4=a_4 \wedge\ldots A_k=a_k \wedge C=c_i) \\ &\quad\quad * P(A_4=a_4 \wedge\ldots A_k=a_k \wedge C=c_i) \end{aligned} \quad (5)$$

In NB, it is assumed that outcome of attribute A_i is independent of the outcome of all other attributes A_j, given c. Thus class conditional probabilities become: $P(A_1=a_1,\ldots A_k=a_k|C=c_i) = \prod_{i=1}^{k} P(A_i=a_i\,|\,C=c_i)$ If the above value is inserted in equation "(4)" it becomes:

$$\equiv \arg\max_{c_i\in C} P(C=c) \prod_{i=1}^{k} P(A_i=a_i\,|\,C=c_i) \quad (6)$$

In Naïve Bayesian learning, the probability values of equation "(6)" are estimated from the given training data. These estimated values are then used to classify unknown examples.

C. Decision Tree Algorithms

The ID3 technique builds decision tree using information theory [29]. The basic strategy used by ID3 is to choose splitting attributes from a data set with the highest information gain. The amount of information associated with an attribute value is related to the probability of occurrence. The concept used to quantify information is called entropy, which is used to measure the amount of randomness from a data set. When all data in a set belong to a single class, there is no uncertainty then the entropy is zero. The objective of decision tree classification is to iteratively partition the given data set into subsets where all elements in each final subset belong to the same class. The entropy calculation is shown in equation "(7)". The value for entropy is between 0 and 1 and reaches a maximum when the probabilities are all the same. Given probabilities $p_1, p_2,..,p_s$ where $\sum_{i=1} p_i=1$,

$$Entropy: H(p_1,p_2,...p_s) = \sum_{i=1}^{s} (p_i \log(1/p_i)) \qquad (7)$$

Given a data set, D, $H(D)$ finds the amount of subset of data set. When that subset is split into s new subsets $S = \{D_1, D_2,...,D_s\}$, we can again look at the entropy of those subsets, A subset of data set is completely ordered if all examples in it are the same class. ID3 chooses the splitting attribute with the highest gain. The ID3 algorithm calculates the gain by the equation "(8)".

$$Gain\ (D,S) = H(D)-\sum_{i=1}^{s} p(D_i)H(D_i) \qquad (8)$$

The C4.5 algorithm improves ID3 through *GainRatio* [30]. For splitting purpose, C4.5 uses the largest GainRatio that ensures a larger than average information gain.

$$GainRatio(D,S) = \frac{Gain(D,S)}{H\left(\frac{|D_1|}{|D|},...,\frac{|D_s|}{|D|}\right)} \qquad (9)$$

The C5.0 algorithm improves the performance of building trees using boosting, which is an approach to combining different classifiers. But boosting does not always help when the training data contains a lot of noise. When C5.0 performs a classification, each classifier is assigned a vote, voting is performed, and the example of data set is assigned to the class with the most number of votes. CART (classification and regression trees) is a process of generating a binary tree for decision making [31]. CART handles missing data and contains a pruning strategy. The SPRINT (Scalable Parallelizable Induction of Decision Trees) algorithm uses an impurity function called *gini* index to find the best split [32]. "Equation (10), defines the *gini* for a data set, *D*".

$$gini\ (D) = \quad 1-\sum p_j^2 \qquad (10)$$

Where, p_j is the frequency of class C_j in D. The goodness of a split of D into subsets D_1 and D_2 is defined by

$$gini_{split}(D) = n_1/n(gini(D_1))+ n_2/n(gini(D_2)) \qquad (11)$$

The split with the best *gini* value is chosen. A number of research projects for optimal feature selection and classification have been done, which adopt hybrid strategy involving evolutionary algorithm and inductive decision tree learning [33], [34], [35], [36].

D. K-Nearest Neighbors

The K nearest neighbors (KNN) is a classification algorithm based on the use of distance measures [37]. It finds k examples in training data that are closest to the test example and assigns the most frequent label among these examples to the new example. When a classification is to be made for a new example, its distance to each attribute in the training data must be determined. Only the K closest examples in the training data are considered further. The new example is then placed in the class that contains the most examples from this set of K closest examples. KNN can be considered decision making technique as equivalent to Bayesian classifier in which the number of neighbors of each example is used as an estimate of the relative posterior probabilities of class membership in the neighborhood of a sample to be classified.

IV. Improved Self Adaptive Bayesian Algorithm

A. Adaptive Bayesian Algorithm

Adaptive Bayesian algorithm creates a function from KDD99 benchmark intrusion detection training data [38], which first estimate the class conditional probabilities for each attribute value based on their frequencies over the weights with match of same class in the training data. In a given training data, $D = \{A_1, A_2,...,A_n\}$ of attributes, where each attribute $A_i = \{A_{i1}, A_{i2},...,A_{ik}\}$ contains attribute values and a set of classes $C = \{C_1, C_2,...,C_n\}$, where each class $C_j = \{C_{j1}, C_{j2},...,C_{jk}\}$ has some values. Each example in the training data contains weight, $W = \{W_1, W_2..., W_n\}$. Initially, all the weights for each example of training data have equal unit value that set to $W_i = 1.0$. Then calculate the sum of weights for each class from the training data by counting how often each class occurs in the training data and the sum of weights for each attribute value with respect to same class in the training data. Next calculate the class conditional probabilities for each attribute value using equation "(12)" from the training data.

$$P(A_{ij}|C_{ji}) = \frac{\sum W_{A_{ij}}}{\sum W_{C_{ji}}} \qquad (12)$$

Here $P(A_{ij}|C_{ji})$ is the class conditional probability, W_A is the sum of weights for each attribute value, and W_C is the sum of weights for each class. After calculating the class conditional probabilities for each attribute value from the training data, the algorithm classify the each test example using equation "(13)".

$$TargetClass = \prod_{Ci} P(A_{ij}|C_{ji}) \qquad (13)$$

If any test example is misclassified, the algorithm updates the weights of training data. The algorithm compare each of test examples with every training example and compute the similarity between them, and then weights of training data are increased by a fixed small value multiplied by the corresponding similarity measure.

$$W_i = W_i + (S*0.01); \quad (14)$$

Here S is the similarity between test and training examples. If the test example is correctly classified, then the weights of training data will remain unchanged. After weights adjustment, the class conditional probabilities for attribute values are recalculated from the modified weights in training data. If the new set of probabilities correctly classifies all the test examples, the algorithm terminates. Otherwise, the iteration continues until all the test examples are correctly classified or the target accuracy is achieved. At this stage the algorithm terminates, the class conditional probabilities are preserved for future classification of seen or unseen examples.

B. Improved Self Adaptive Bayesian Algorithm

Improved self adaptive Bayesian algorithm (ISABA) is the modification of adaptive Bayesian algorithm. Given a training data ISABA initializes the weights of each example W_i set to 1.0 and estimates the prior probability *P(Cj)* for each class by summing the weights how often each class occurs in the training data. For each attribute in training data, A_i the number of occurrences of each attribute value A_{ij} can be counted by summing the weights to determine the probability $P(A_{ij})$. Similarly, the probability $P(A_{ij} \mid C_j)$ can be estimate by summing the weights how often each attribute value occurs in the class in the training data. An example in the training data may have many different attributes $A = \{A_1, A_2,\ldots,A_n\}$, and each attribute have many values $A_i = \{A_{i1}, A_{i2},\ldots,A_{ik}\}$. The conditional probability $P(A_{ij} \mid C_j)$ estimate for all values of attributes. Then the algorithm uses these conditional probabilities to classify all the training examples. When classifying a training example, the conditional and prior probabilities generated from the training data are used to make the prediction. This is done by multiplying the probabilities of the different attribute values from the example. Suppose the training example e_i has p independent attribute values $\{A_{i1}, A_{i2},\ldots,A_{ip}\}$, the algorithm has $P(A_{ik} \mid C_j)$, for each class C_j and attribute A_{ik}, and then estimate probability $P(e_i \mid C_j)$ by using equation “(15)”.

$$P(e_i \mid C_j) = P(C_j) \prod_{k=1\to p} P(A_{ij} \mid C_j) \quad (15)$$

To calculate the probability $P(e_i)$, the algorithm estimate the likelihood that e_i is in each class. The probability that e_i is in a class is the product of the conditional probabilities for each attribute values. The posterior probability $P(C_j \mid e_i)$ is then found for each class. The class with the highest probability is the one chosen for the example in training data. Now the algorithm updates the weights for each example in the training data with the highest value of posterior probability $P(C_j \mid e_i)$ for that example and also changes the class value associate with highest posterior probability. If any example in the training data is misclassified then the algorithm again calculates the prior probability *P(Cj)* and conditional probability $P(A_{ij} \mid C_j)$ from the training examples using the updated weights, and again classify the training examples and updates the weights of training examples. This iteration will continue until all the training examples are correctly classified or the target accuracy is achieved.

After classifying the training examples, the algorithm classify the test examples using the conditional probabilities $P(A_{ij} \mid C_j)$. If any test example is misclassified, the algorithm again updates the weights of training examples. The algorithm compares each of test example with every training examples and compute the similarity between them (out of four attributes two attribute values are same then similarity becomes 0.5 or one attribute value is same then similarity is 0.25 and so on), and then weights of training examples are increased by a fixed small value multiplied by the corresponding similarity measure. If the test examples are correctly classified, then the weights of training examples will remain unchanged. After weights adjustment, the conditional probabilities $P(A_{ij} \mid C_j)$ for attribute values are recalculated from the modified weights of training examples. If the new set of probabilities correctly classifies all the test examples, the algorithm stores the conditional probabilities and builds a decision tree by information gain using last updated weights. Otherwise, the iteration continues until all the test examples are correctly classified or the target accuracy is achieved. At this stage the algorithm correctly classifies all the test examples, the conditional probabilities $P(A_{ij} \mid C_j)$ are preserved for future classification of seen or unseen intrusions, and builds a decision tree by information gain using the last updated weights of training examples, which is also used for classification of seen or unseen intrusions. The main procedure of ISABA algorithm is described as follows.

Algorithm ISABA

Input: training data *D*, testing data

Output: intrusion detection model

Procedure:

1. Initialize all the weights in *D*, W_i=1.0.
2. Calculate the prior probabilities $P(C_j)$ for each class C_j from *D*.

$$P(C_j) = \frac{\sum_{Ci} W_i}{\sum_{i=1}^{n} W_i}$$

3. Calculate the probabilities $P(A_{ij})$ for each attribute value A_{ij} from *D*.

$$P(A_{ij}) = \sum_{A_{ij}\to C_j} W_i$$

4. Calculate the conditional probabilities $P(A_{ij} \mid C_j)$ for each attribute values from *D*.

$$P(A_{ij} \mid C_j) = \frac{P(A_{ij})}{\sum_{C_i} W_i}$$

5. Classify each example in *D*.

$$P(e_i \mid C_j) = P(C_j) \prod P(A_{ij} \mid C_j)$$

6. Initialize all the weights in *D* with Maximum Likelihood (ML) of posterior probability $P(C_j|e_i)$ and change the class value associated with highest posterior probability.

$$W_i = P_{ML}(C_j|e_i)$$

$$C_j = C_i \rightarrow P_{ML}(C_j|e_i)$$

7. If any example in *D* is misclassified, then go to step 2, else go to step 8.
8. Classify all test examples with the conditional probabilities $P(A_{ij} \mid C_j)$.
9. If any test example is misclassified then update the weights of *D* using similarity *S*.

$$W_i = W_i + (S+0.01)$$

10. If weight update, then recalculate the probabilities $P(A_{ij})$ and $P(A_{ij} \mid C_j)$ using updated weights of *D* and go to step 8.
11. If all test examples are correctly classified then the algorithm stores conditional probabilities for future classification of seen or unseen intrusions.
12. The algorithm builds decision tree by information gain using final updated weights of training examples.

V. Experimental Results

A. Experimental Data

The KDD99 dataset is a common benchmark for evaluation of intrusion detection techniques [39]. In the 1998 DARPA intrusion detection evaluation program, a simulated environment was set up to acquire raw TCP/IP dump data for a local-area network (LAN) by the MIT Lincoln Lab to compare the performance of various intrusion detection methods. The DARPA98 was operated like a real environment, but being blasted with multiple intrusion attacks and received much attention in the research community of adaptive intrusion detection. In 1999, based on the DARPA98 data the Third International Knowledge Discovery and Data Mining Tools Competition established the KDD99 benchmark dataset for intrusion detection based on data mining. In the KDD99 data set, each data example represents attribute values of a class in the network data flow, and each class is labeled either as normal or as an attack with exactly one specific attack type. The KDD99 data records are all labeled with one of the following five types:

1) Normal connections are generated by simulated daily user behavior such as downloading files, visiting web pages, etc.

2) Denial of Service (DoS) attack causes the computing power or memory of a victim machine too busy or too full to handle legitimate requests. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users. Example of DoS attacks are Apache2, Land, Mail bomb, Back, etc.

3) User to Root (U2R) is a class of attacks that an intruder/hacker begins with the access of a normal user account and then becomes a super-user by exploiting various vulnerabilities of the system. Most common exploits of U2R attacks are regular buffer overflows, Loadmodule, Fdformat, and Ffbconfig.

4) Remote to User (R2L) is a class of attacks that a remote user gains access of a local account by sending packets to a machine over a network communication, which include Sendmail, and Xlock.

5) Probing (Probe) is an attack scans a network to gather information or find known vulnerabilities. An intruder with a map of machines and services that are available on a network can use the information to look for exploits.

B. Eeperimental Analysis

Correct classification of known and unknown intrusions is one of the central problems for network-based intrusion detection. Many supervised and unsupervised learning algorithms already applied to classifying intrusions but their performance were not very satisfactory due to the challenging problem of detection novel attacks with low false alarms. In order to evaluate the performance of improved self adaptive Bayesian algorithm (ISABA) for intrusion detection, we performed 5-class classification using KDD99 benchmark dataset. The training and testing data are taken randomly from the KDD99 dataset with different ratios of positive versus negative instance. The training data are used to train the algorithms, and the test data are used to evaluate the performance of the algorithms. Table I shows the number of examples in 10% training data and 10% testing data of KDD99 dataset. There are some new attack examples in testing data, which is no present in the training dataset.

TABLE I.
NUMBER OF EXAMPLES IN TRAINING AND TESTING DATA

Attack Types	Training Examples	Testing Examples
Normal	97277	60592
Probing	4107	4166
Denial of Service	391458	237594
User to Root	52	70
Remote to User	1126	8606
Total Examples	494020	311028

In the experiments, first we performed the classification on KDD99 testing data using naïve Bayesian classifier (NB), adaptive Bayesian algorithm (ABA), and improved self adaptive Bayesian algorithm (ISABA) and compare their results shown in Table II. It is clear that the proposed new algorithm is approximately 2 times faster in training and testing than conventional algorithms.

TABLE II.
TRAINING AND TESTING TIME COMPARISON

	Training Time (s)	Testing Time (s)
Naïve Bayesian Classifier	42.9	18.4
Adaptive Bayesian Algorithm	24.6	9.6
Improved Self Adaptive Bayesian Algorithm	28.4	7.5

The performance comparison between NB and ISABA on KDD99 (using 41 attributes) testing data is listed in Table III, which shows that the ISABA performs balanced and high classification rates on 5 attack classes of KDD99 data and minimize the false positives.

TABLE III.
PERFORMANCE COMPARISON BETWEEN NB AND ISABA

	Normal	Probe	DoS	U2R	R2L
Naïve Bayesian Classifier (DR %)	99.25	99.13	99.69	64	99.11
Naïve Bayesian Classifier (FP %)	0.08	0.45	0.04	0.14	8.02
Improved Self Adaptive Bayesian Algorithm (DR %)	99.62	99.22	99.49	99.17	99.15
Improved Self Adaptive Bayesian Algorithm (FP %)	0.05	0.36	0.03	0.10	6.91

We tested our proposed algorithm using the reduced dataset of 12 attributes and 17 attributes in KDD99 dataset, which increase the classification rate for intrusion classes that are summarized in Table IV.

TABLE IV.
PERFORMANCE OF ISABA USING REDUCED DATASETS

	12 Attributes	17 Attributes
Normal	99.97	99.96
Probe	99.91	99.95
DoS	99.99	99.98
U2R	99,36	99.46
R2L	99.53	99.69

We also compare the detection performance among Support Vector Machines (SVM), Neural Network (NN), Genetic Algorithm (GA), Naïve Bayesian Classifier (NB), and improved self adaptive Bayesian algorithm (ISABA) on KDD99 dataset [40],[41],[42]. In total, 40 attributes of KDD99 dataset have been used. Each connection can be categorized into five main classes (one normal class and four main intrusion classes: probe, DoS, U2R, and R2L). The experimental setting uses 494020 data samples for training and 311028 data samples for testing. The comparative results are summarized in Table V.

TABLE V.
COMPARISON OF SEVERAL ALGORITHMS

	SVM	NN	GA	NB	ISABA
Normal	99.4	99.6	99.3	99.55	99.82
Probe	89.2	92.7	98.46	99.43	99.72
DoS	94.7	97.5	99.57	99.69	99.49
U2R	71.4	48	99.22	64	99.47
R2L	87.2	98	98.54	99.11	99.35

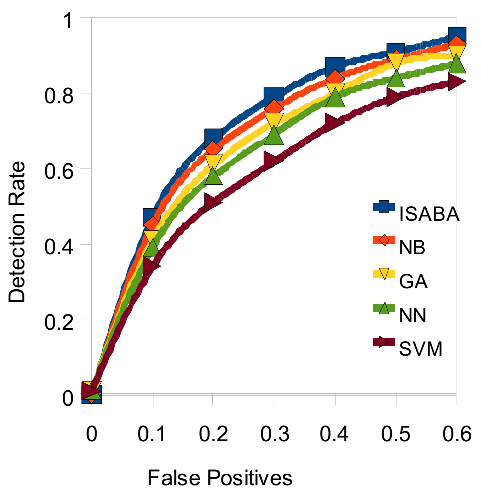

Figure 1. ROC curves for alert classification systems.

The ROC (relative operating characteristic) curve shows the relationship between detection rate and false positives on 10% KDD99 testing data in figure 1.

VI. CONCLUSION AND FUTURE WORKS

The main advantage of this proposed algorithm is to generate a minimal rule set for network intrusion detection, which can detect network intrusions based on previous activities. Proposed algorithm analyzes the large volume of network data and considers the complex properties of attack behaviors to improve the performance of detection speed and detection accuracy. This paper presents a new intrusion detection algorithm based on intelligent machine learning algorithms. In this paper we have concentrated on the development of the performance of naïve Bayesian classifier, which adjusts the weights of training examples until either all the test examples are correctly classified or the target accuracy on test examples is achieved. The experimental results marked that this algorithm minimized false positives, as well as maximize balance detection (classification rates) on the 5 classes of KDD99 data. The future research issues will be applying domain knowledge of security to improve the detection accuracy, and visualizing the procedure of intrusion detection in real world problem domains.

ACKNOWLEDGMENT

Support for this research received from Ministry of Science and Information & Communication Technology, Government of Bangladesh. We would like to thank Prof. Dr. Chowdhury Mofizur Rahman, United International University, Bangladesh for fruitful discussions and valuable help in the implementation of this algorithm.

REFERENCES

[1] James P. Anderson, "Computer security threat monitoring and surveillance," Technical report, James P. Anderson Co., Fort Washington, Pennsylvania, April 1980.

[2] Cannady J., "Artificial neural networks for misuse detection," Proceedings of the '98 National Information System Security Conference (NISSC'98), Arlington: Virginia Press, 1998, pp. 443-456.

[3] Shon T, Seo J, and Moon J, "SVM approach with a genetic algorithm for network intrusion detection," Proceedings of the 20th International Symposium on Computer and Information Sciences (ISCIS 05), Berlin: Springer Verlag, 2005, pp. 224-233.

[4] Yu Y, and Huang Hao, "An ensemble approach to intrusion detection based on improved multi-objective genetic algorithm," Journal of Software, Vol.18, No.6, June 2007, pp.1369-1378.

[5] J. Luo, and S. M. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," International Journal of Intelligent Systems, 2000, pp. 687-703.

[6] W.K. Lee, and S. J. Stolfo, "A data mining framework for building intrusion detection model," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA: IEEE Computer Society Press, 1999, pp. 120-132.

[7] Kononenko I, "Comparison of inductive and naïve Bayesian learning approaches to automatic knowledge acquistion," in Wieling, B. (Ed), Current trend in knowledge acquistion, Amsterdam, IOS press, 1990.

[8] Langely, P., Iba, W., Thomas, and K., "An analysis of Bayesian classifier," in Proceedings of the 10th national Conference on Artificial Intelligence (San Matro, CA: AAAI press) 1992, pp. 223-228.

[9] Denning, and Dorothy E., "An Intrusion Detection Model," Proceedings of the Seventh IEEE Symposium on Security and Privacy, May 1986, pp. 119-131.

[10] Sebring, Michael M., Whitehurst, and R. Alan., "Expert Systems in Intrusion Detection: A Case Study," The 11th National Computer Security Conference, October, 1988.

[11] Smaha, and Stephen E., "Haystack: An Intrusion Detection System," The 4th Aerospace Computer Security Applications Conference, Orlando, FL, December, 1988.

[12] Vaccaro, H.S., and Liepins, G.E., "Detection of Anomalus Computer Session Activity," The 1989 IEEE Symposium on Security and Privacy, May, 1989.

[13] Heberlein, and Let al., "A Network Secutiry Mnitor," Proceedings of the IEEE Computer Society Symposium, Research in Security and Privacy, May 1990, pp. 296-303.

[14] Lunt, and Teresa F., "IDES: An Intelligent System for Detecting Intruders," Proceedings of the Symposium on Computer Security; Threats, and Countermeasures; Rome, Italy, November 22-23, 1090, pp.110-121.

[15] Teng, Henry S., Chen, Kaihu, Lu, and Stephen C-Y, "Adaptive Real-time Anomaly Detection using Inductively Generated Sequential Patterns," 1990 IEEE Symposium on Recearch in Security and Privacy, Oakland, CA, pp. 296-304.

[16] Snapp, Steven R, and Ho., "DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and An Early Prototype," The 14th National Computer Security Conference, October, 1991, pp.167-176.

[17] Jackson, Kathleen, DuBois, David H., Stallings, and Cathy A., "A Phased Approach to Network Intrusion Detection," 14th National Computing Security Conference, 1991.

[18] Lunt, and Teresa F., "Detecting Intrusion in Computer Systems," 1993 Conference on Auditing and Computer Technology, SRI International.

[19] Paxson, and Vern, "Bro: A System for Detecting Network Intruders in Real-Time," Proceedings of The 7th USENIX Security Symposium, San Antonio, TX, 1998.

[20] Barbara, Daniel, Couto, Julia, Jajodia, Sushil, Popyack, Leonard, Wu, and Ningning, "ADAM: Detecting Intrusion by Data Mining," Proceedings of the IEEE Workshop on Information Assurance and Security, West Point, New York June 5-6, 2001.

[21] Jackson, T., Levine, J., Grizzard, J., Owen, and H., "An investigation of a compromised host on a honeynet being used to increase the security of a large enterprise network," Proceedings of the 2004 IEEE Workshop on Information Assurance and Security, IEEE, 2004.

[22] Krasser, S., Grizzard, J., Owen, H., and Levine. J., "The use of honeynets to increase computer network security and user awareness," Journal of Security Education, Volume 1, pp. 23-37, 2005.

[23] Thomas H. Ptacek, and Timothy N. Newsham, "Insertion, evasion and denial of service: Eluding network intrusion detection," Technical report, Secure Networks Inc., 1998.

[24] D.Y. Yeung, and Y.X. Ding, "Host-based intrusion detection using dynamic and static behavioral model", Pattern Recognition, 36, pp. 229-243, 2003.

[25] SunSoft. SunSHIELD Basic Security Model. SunSoft. 1995.

[26] Diego Zamboni. "Using Internal Sensors for Computer Intrusion Detection," PhD thesis, Purdue University, 2001.

[27] Tadeusz Pietraszek, and Chris Vanden Berghe. "Defending against injection attacks through context-sensitive string evaluation", In Recent Advances in Intrusion Detection (RAID2005), volume 3858 of Lecture Notes in Computer Science, pp. 124-145, Seattle, WA, 2005. Springer-Verlag.

[28] Sebastiaan Tesink, "Improving Intrusion Detection System throung Machine Learning," Technical Report Series no. 07-02, ILK Research Group, Tilburg University, March 2007.

[29] J. R. Quinlan, "Induction of Decision Tree," Machine Learning Vol. 1, pp. 81-106, 1986.

[30] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[31] L. Breiman, J. H. Friedman, R. A. Olshen and C.J. Stone, "Classification and Regression Trees," Statistics probability series, Wadsworth, Belmont, 1984.

[32] John Shafer, Rakesh Agarwal, and Manish Mehta, "SPRINT: A Scalable Parallel Classifier for Data Maining," in Proceedings of the VLDB Conference, Bombay, India, September 1996.

[33] D. Turney, "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm," Journal of Artificial Intelligence Research, pp. 369-409, 1995.

[34] J. Bala, K. De Jong, J. Haung, H. Vafaie and H. Wechsler, "Hybrid Learning using Genetic Algorithms and Decision Trees for Pattern Classification," in Proceedings of 14th International Conference on Artificial Intelligence, 1995.

[35] C. Guerra-Salcedo, S. Chen, D. Whitley, and Stephen Smith, "Fast and Accurate Feature Selection using Hybrid Genetic Strategies," in Proceedings of the Genetic and Evolutionary Computation Conference, 1999.

[36] S. R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology, " IEEE Transactions on Systems, Man and Cybernetics 21(3), pp. 660-674, 1991.

[37] Duda, R., P.E. Hart, and D.G. Stork, "Pattern classification," Second edn. John Wiley & Sons, 2001.

[38] Dewan Md. Farid, and Mohammad Zahidur Rahman, "Learning Intrusion Detection Based on Adaptive Bayesian Algorithm," Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008), Khulna, Bangladesh, 25-27 December, 2008, pp.652-656.

[39] The KDD Archive. KDD99 cup dataset. 1999. http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

[40] Mukkamala S, Sung AH, and Abraham A, "Intrusion dection using an ensemble of intelligent paradigms,"

Proceedings of Journal of Network and Computer Applications, 2005, 2(8): pp. 167-182.
[41] Chebrolu S, Abraham A, and Thomas JP, “Feature deduction and ensemble design of intrusion detection systems.” Computer & Security, 2004, 24(4), pp. 295-307.
[42] Lee WK, Stolfo SJ, and Mok KW, “A data mining framework for building intrusion detection models,” Proceedings of the ’99 IEEE Symp. On Security and Privacy, Oakland: IEEE Computer Society. 1999, pp. 120-132.

Dewan Md. Farid was born in 1979. He received the B.Sc. Engineering in Computer Science and Engineering from Asian University of Bangladesh in 2003 and Master of Science in Computer Science and Engineering from United International University, Bangladesh in 2004. He is continuing Ph.D. at Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh. His major field of study is artificial intelligence, machine leaning, and data mining.

He is a faculty member at Department of Computer Science and Engineering, United International University, Bangladesh. He has published two conference papers, which include Learning Intrusion Detection Based on Adaptive Bayesian Algorithm, etc.

Mr. Farid is a member of Bangladesh Computer Society and Research Scholar Association, Jahangirnagar University. In 2008, he received the Fellowship of National Science and Information & Communication Technology (NSICT) from Ministry of Science and Information & Communication Technology, Government of Bangladesh.

Mohammad Zahidur Rahma is currently a Professor at Department of Computer Science and Engineering, Jahangirnager University, Banglasesh. He obtained his B.Sc. Engineering in Electrical and Electronics from Bangladesh University of Engineering and Technology in 1986 and his M.Sc. Engineering in Computer Science and Engineering from the same institute in 1989. He obtained his Ph.D. degree in Computer Science and Information Technology from University of Malaya in 2001. He is a co-author of a book on E-commerce published from Malaysia. His current research includes the development of a secure distributed computing environment and e-commerce.