Adaptive Intrusion Detection based on Boosting and Naïve Bayesian Classifier

Dewan Md. Farid
Department of CSE
Jahangirnagar University
Dhaka-1342, Bangladesh

Mohammad Zahidur Rahman
Department of CSE
Jahangirnagar University
Dhaka-1342, Bangladesh

Chowdhury Mofizur Rahman
Department of CSE
United International University
Dhaka-1209, Bangladesh

ABSTRACT
In this paper, we introduce a new learning algorithm for adaptive intrusion detection using boosting and naïve Bayesian classifier, which considers a series of classifiers and combines the votes of each individual classifier for classifying an unknown or known example. The proposed algorithm generates the probability set for each round using naïve Bayesian classifier and updates the weights of training examples based on the misclassification error rate that produced by the training examples in each round. This algorithm addresses the problem of classifying the large intrusion detection dataset, which improves the detection rates (DR) and reduces the false positives (FP) at acceptable level in intrusion detection. We tested the performance of the proposed algorithm with existing data mining algorithms by employing on the KDD99 benchmark intrusion detection dataset, and the experimental results proved that the proposed algorithm achieved high detection rates and significantly reduced the number of false positives for different types of network intrusions.

Keywords
Boosting, Naïve Bayesian Classifier, Intrusion Detection, Detection Rate, False Positive.

1. INTRODUCTION
Due to the large volumes of intrusion detection dataset, the intelligent computing society has been applied many data mining algorithms for detecting intrusions in the last decades [1]-[3]. Today's real-world intrusion detection datasets are complex, dynamic, and composed of many different attributes, which are highly susceptible to noise, missing and inconsistent data due to their typically huge size. The task of data mining and knowledge discovery from data (KDD) based intrusion detection systems (IDS) is to find the interesting hidden intrusion patterns in large intrusion detection dataset and realizing the performance optimization of detection rules. Intrusion detection system (IDS) is the combination of both hardware and software that is used to detect intrusions or attacks in computer system or network, and then notifies intrusion prevention system (IPS) or network security administrator about the intrusions. However, currently available commercial IDS are misuse based, which can only detect known intrusions with very low false positives that are already stored in the dataset, but now-a-days intruders are very intelligent and they frequently change the intrusion patterns to attack into the network. Intruders are classified into two categories like inside and outside intruders. Inside intruders are the valid user of the network and have some authority, but seek to gain additional ability to take actions without legitimate authorization. On the other side, outside intruders do not have any authorized access to the network that they attack. Ideally, IDS should have an attack detection rate of 100% along with false positive rate 0%, which is really very hard to achieve.

Today, data mining have become an indispensable tools for analyzing the large volumes of intrusion detection data to detect intrusions by finding the hidden intrusion patterns from the dataset [4]-[7]. The naïve Bayesian (NB) classifier is an efficient and well known technique for performing classification task in data mining, which is widely applied in many real world applications including intrusion detection problem [8]-[16]. The NB classifier provides an optimal way to predict the class of an unknown example whose attribute values are known, but class value is unknown by calculating prior and conditional probabilities from the training dataset. Boosting is the process of combining many classifiers to generate a single strong classifier with very low error. In this paper, we present a new learning algorithm based on boosting, and naïve Bayesian classifier for adaptive intrusion detection. The proposed algorithm first initializes the weight of training examples to $1/n$, where n is the total number of examples in training dataset, and then creates a new dataset from training dataset using selection with replacement technique. After that it calculates the prior and conditional probabilities of new dataset, and classifies the training examples with these probabilities value. The weights of the training examples updated according to how they were classified. If a training example is misclassified then its weight is increased, or if correctly classified then its weight is decreased. Then the algorithm creates another new data set with the misclassification error produced by each training example from training dataset, and continues the process until all the training examples are correctly classified. To classify a new example use all the probabilities in each round (each round is considered as a classifier) and consider the class of new example with highest classifier's vote. The proposed algorithm has been successfully tested on the KDD99 benchmark network intrusion detection dataset from UCI machine learning repository, which achieved high detection rates with very low false positives.

The remainder of this paper is organized as follows. In Section 2, we describe the data mining for intrusion detection overview, and related works. In Section 3, we present the boosting, naïve Bayesian classifier, and the proposed learning algorithm. In Section 4, we apply the proposed algorithm to the area of intrusion detection using KDD99 benchmark network intrusion detection dataset, and compare the results with other related algorithms. Finally, Section 5 contains the conclusions with future works.

2. INTRUSION DETECTION

An intrusion can be defined as any set of actions that threaten the integrity, confidentiality, or availability of computational resources of a computer system or network. Intrusion detection have become a critical component of network administration as the extensive growth of the Internet and the tools for intruding and attacking networks are available now-a-days. An intrusion detection system (IDS) for a large complex network can typically generate thousands or millions of alarms per day, representing an overwhelming task for the security analysis.

2.1 Host vs. Network IDS

Initially IDS was developed for host-based computer systems. The host-based IDS (HIDS) are located in the server computers and examine the internal interfaces [17]. It detects intrusions by analyzing application logs, system calls, file-system modifications, and other host activities that related to the server computers. Context Sensitive String Evaluation (CSSE) is one of the Host-based IDS for detecting intrusions in applications with extremely low false-positives [18]. CSSE uses an instrumented execution environment (such as PHP or Java Virtual Machine) and therefore has access to all necessary contexts required to detect and more importantly prevent attacks. The context is provided by the metadata, which describes the fragments of the output expression that requires checking and examining the intercepted call to the API function. CSSE uses contextual information to check the unsafe fragments for syntactic content. Depending on the mode of CSSE it can raise an alert and prevent the execution of the dangerous content (both intrusion detection and prevention). Currently CSSE is available as research-prototype IDS for the PHP platform [19], [20].

With the popularization of computer networks the idea of IDS gradually shifted toward the network-based IDS. It monitors and analyzes network packets to detect intrusions in the network [21]. Snort is an open source network intrusion detection and prevention system (NIDPS) capable of performing packet logging and real-time traffic analysis of IP networks. Snort was written by Martin Roesch and is now developed by Sourcefire [22], [23]. Snort performs protocol analysis, content searching/matching, and is commonly used to actively block a variety of attacks. Most of the current attacks happen at higher layers: transport (TCP/UDP) or application (HTTP, RPC) layers and Snort uses so-called preprocessors which perform stream reassembly and normalization of higher-level protocols. To detect an attack targeting a web server the preprocessors normalize the IP-level traffic, TCP state machine emulation and stream reassembly, HTTP-level normalization, defragmentation, and Unicode decoding.

2.2 Misuse vs. Anomaly IDS

Intrusion detection model is broadly classified into two categories: misuse-based and anomaly-based intrusion detection model.

Misuse-based IDS are also known as signature-based or pattern-based IDS, which detect known intrusions based on the attacks that stored in database with very low false positives. It performs pattern matching of incoming packets and/or command sequences to the signatures of known attacks. The detection rate of misuse-based IDS is relatively low, because the attacker always tries to modify the basic attack signature in such a way that will not match the attack signature, which is already installed in the database. It can protect the computer system/network immediately upon installation, but it requires frequently signature updates to keep the signature database up-to-date. Misuse-based IDS use various techniques including rule-based expert systems, model-based reasoning systems, state transition analysis, genetic algorithms, and fuzzy logic.

Anomaly-based IDS can detect known or unknown intrusions by detecting deviations from normal behaviors. It creates a profile from normal behaviors and then any activities that deviated from this profile are treated as a possible intrusion. Many data mining algorithms already been used for anomaly detection such as decision tree (DT), naïve Bayesian (NB), neural networks (NN), support vector machines (SVM), and Principal Components Analysis (PCA) etc. The major drawback of anomaly-based IDS is to provide a large number of false positives.

2.3 Related Work

Intrusion detection was first introduced by James P. Anderson in 1980 by introducing a threat classification model that develops a security monitoring surveillance system based on detecting anomalies in user behaviors [24]. Later in 1987, Dr. Denning proposed several models for IDS based on statistics, Markov chains, time-series, etc [25]. In 1988, a statistical anomaly-based IDS was proposed by Haystack [26], which used both user and group-based anomaly detection strategies. In 2005, Fan et al. proposed a method that injects artificial anomaly data into the training data to detect known and unknown intrusions, which help a baseline classifier to distinguish between normal and anomalous data [27]. In 2006, Bouzida et al. applied decision tree (DT) for anomaly-based intrusion detection, which assigns a default class to the test instance that is not covered by the tree and then the default class are examined for unknown attack analysis [28]. In 2004, Peddabachigari et al. [29] applied decision tree (DT) and support vector machine (SVM) for intrusion detection, which proved that DT is better than SVM in terms of overall accuracy. Particularly, DT much better in detecting user to root (U2R) and remote to local (R2L) network attacks, compared to SVM. In 2001, Barbara et al. [30] proposed a method for detecting new attacks and reducing false positives, which estimates the probability using Bayes estimators to enhance the ability of ADAM based IDS [31]. In 2004, Amor et al. performed an experimental analysis to compare the performance between NB classifier and DT classifier by employing KDD99 dataset, and the result proved that NB classifier is 7 times faster than DT with respect to running time, and DT outperforms in classifying normal, denial of service (DoS), and remote to local (R2L) attacks, whereas NB classifier is superior in classifying Probing and user to root (U2R) attacks [32]. In 2007, Panda and Patra [33] proposed a method using naïve Bayes to detect signatures of specific attacks. They used KDD99 dataset for experiment, and the authors give a conclusion that NB classifier performs back propagation neural network classifier in terms of detection rates and false positives. It is also reported that NB classifier produces a relatively high false positive. Later in 2009, the same authors Panda and Patra [34] compares NB classifier with 5 other similar classifiers, i.e., JRip, Ridor, NNge, Decision Table, and Hybrid Decision Table, and experimental results shows that the NB classifier is better than other classifiers.

2.4 Data Mining for Intrusion Detection

As current intrusion detection systems (IDS) have many limitations, the data mining for intrusion detection open a new research area in intelligent computing. Data mining algorithms can be used for misuse detection and anomaly detection. In misuse detection, the training data are labeled as either “normal” or “intrusion,” and then a classifier detect the known intrusions. Anomaly detection builds models of normal behavior and automatically detects significant deviations from it. Fig 1 shows the architecture of data mining based IDS.

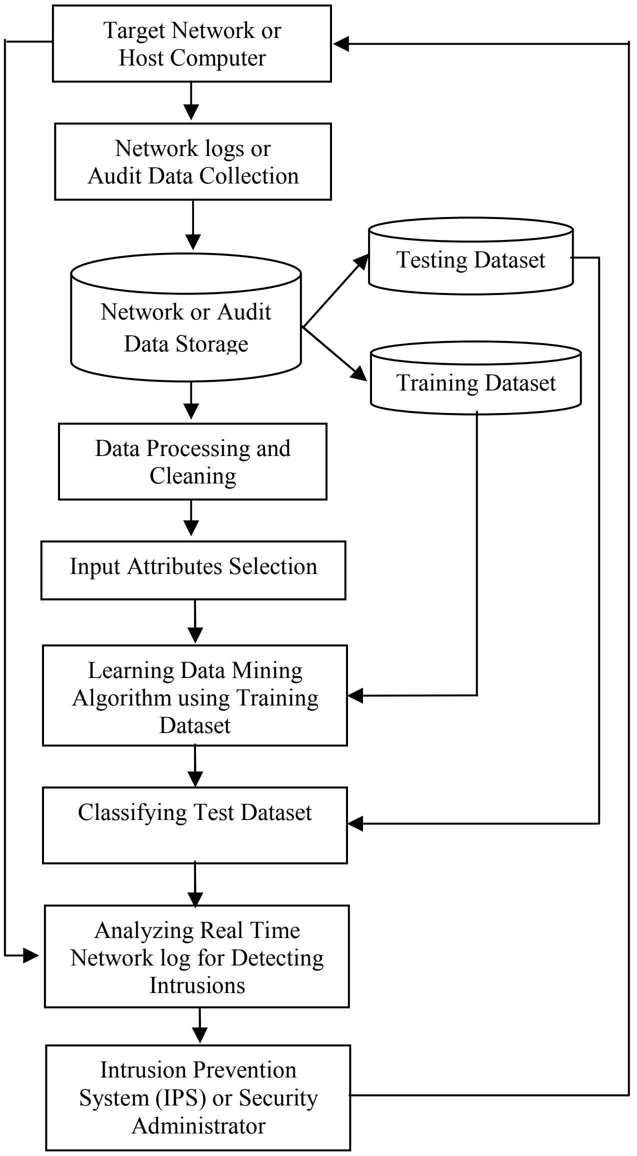
Target Network or Host Computer
Network logs or Audit Data Collection
Network or Audit Data Storage
Testing Dataset
Training Dataset
Data Processing and Cleaning
Input Attributes Selection
Learning Data Mining Algorithm using Training Dataset
Classifying Test Dataset
Analyzing Real Time Network log for Detecting Intrusions
Intrusion Prevention System (IPS) or Security Administrator

Fig 1: Architecture of Data Mining based IDS.

Data mining based IDS collects audit data or network logs from computer system or network, and then applies intelligent methods to extract hidden intrusion patterns from the data. It first collects the audit data from the network using multiple sensors, and stores the audit data for future reference. Training and testing datasets are generated from the collected audit data. Data processing and cleaning is the process of removing noise and inconsistent data from dataset and formatted the dataset that is suitable for mining. Intrusion detecting dataset may contain hundreds of input attributes, and many of which may be irrelevant to the mining task or redundant, because the information they added is contained in other attribute. Input attribute selection reduces the dataset size by removing irrelevant or redundant attributes. The use of all attributes may simply increase the overall complexity of detection model, increase computational time, and decrease the detection accuracy of the intrusion detection algorithms. It has been tested that effective attributes selection improves the detection rates for different types of network intrusions in intrusion detection. After attribute selection the data mining algorithm is trained by the examples from the training dataset and then classifies the examples of testing dataset. When the rules are generated then the model classifies the real time network data and notifies intrusion prevention system (IPS) or security administrator about the intrusions in the network. IPS or security administrator carries out the prescriptions controlled by the IDS.

3. LEARNING ALGORITHMS

3.1 Boosting

Boosting is an iterative process, which adaptively changes the distribution of training examples so that the base classifiers will focus on examples that are hard to classify. The concept of adaptive boosting called AdaBoost algorithm was first introduced by Freund and Schapire in 1997 [35] that classify an example by voting the weighted predictions of a set of base classifiers, which are generated in a series of rounds. The major drawback of boosting is overfitting; that is, with many rounds of boosting, the test error increase as the final classifier becomes overly complex. Boosting have become one of the alternative framework for classifier design, together with the more established classifiers, like Bayesian classifier, decision tree, neural network, and support vector machine. Boosting assigns a weight to each training example and adaptively changes the weight at the end of each boosting round. A sample is drawn according to the sampling distribution of the training examples to obtain a new training dataset. Next, a classifier is induced from the training dataset and used to classify all the examples in the original dataset. The weights of the training examples are updated at the end of each boosting round. Examples that are misclassified will have their weights increased, while those that are correctly classified will have their weight decreased. This forces the classifier to focus on examples that are difficult to classify in subsequent iterations.

3.2 Naïve Bayesian Classifier

Naïve Bayesian classifier is a simple classification scheme, which estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label *c*. The conditional independence assumption can be formally stated as follows:

$$P(A \mid C = c) = \prod_{i=1}^{n} P(A_i \mid C = c) \qquad (1)$$

Where each attribute set $A = \{A_1, A_2, \ldots, A_n\}$ consists of *n* attribute values. With the conditional independence assumption, instead of computing the class-conditional probability for every combination of A, only estimate the conditional probability of each A_i, given C. The latter approach is more practical because it does not require a very large training set to obtain a good estimate of the probability. To classify a test example, the naïve Bayesian classifier computes the posterior probability for each class C.

$$P(C \mid A) = \frac{P(C)\prod_{i=1}^{n} P(A_i \mid C)}{P(A)} \tag{2}$$

Since $P(A)$ is fixed for every A, it is sufficient to choose the class that maximizes the numerator term,

$$P(C)\prod_{i=1}^{n} P(A_i \mid C) \tag{3}$$

The naïve Bayesian classifier has several advantages. It is easy to use, and unlike other classification approaches, only one scan of the training data is required. The naïve Bayesian classifier can easily handle missing attribute values by simply omitting the probability when calculating the likelihoods of membership in each class. The NB classifier is straightforward to use, where there are simple relationships, it often does yield good results.

3.3 Proposed Leaning Algorithm

Given a training data $D = \{t_1,\ldots,t_n\}$, where $t_i = \{t_{i1},\ldots,t_{ih}\}$ and the attributes $\{A_1, A_2,\ldots,A_n\}$. Each attribute A_i contains the following attribute values $\{A_{i1}, A_{i2},\ldots,A_{ih}\}$. The training data D also contains a set of classes $C = \{C_1, C_2,\ldots,C_m\}$. Each training example has a particular class C_j. The algorithm first initializes the weights of training examples to an equal value of w_i=1/n, where n is the total number of training examples in D. Then the algorithm generates a new dataset D_i with equal number of examples from training data D using selection with replacement technique and calculates the prior $P(C_j)$ and class conditional $P(A_{ij}|C_j)$ probabilities for new dataset D_i.

The prior probability $P(C_j)$ for each class is estimated by counting how often each class occurs in the dataset D_i. For each attribute A_i the number of occurrences of each attribute value A_{ij} can be counted to determine $P(A_i)$. Similarly, the class conditional probability $P(A_{ij}|C_j)$ for each attribute values A_{ij} can be estimated by counting how often each attribute value occurs in the class in the dataset D_i. Then the algorithm classifies all the training examples in training data D with these prior $P(C_j)$ and class conditional $P(A_{ij}|C_j)$ probabilities from dataset D_i. For classifying the examples, the prior and conditional probabilities are used to make the prediction. This is done by combining the effects of the different attribute values from that example. Suppose the example e_i has independent attribute values $\{A_{i1}, A_{i2},\ldots,A_{ip}\}$, we know $P(A_{ik} \mid C_j)$, for each class C_j and attribute A_{ik}. We then estimate $P(e_i \mid C_j)$ by

$$P(e_i \mid C_j) = P(C_j) \prod_{k=1\to p} P(A_{ij} \mid C_j) \tag{4}$$

To classify the example, the probability that e_i is in a class is the product of the conditional probabilities for each attribute value with prior probability for that class. The posterior probability $P(C_j \mid e_i)$ is then found for each class and the example classifies with the highest posterior probability value for that example. The algorithm classifies each example t_i □ D with maximum posterior probability. After that the weights of the training examples t_i in training data D are adjusted/ updated according to how they were classified. If an example was misclassified then its weight is increased, or if an example was correctly classified then its weight is decreased.

To updates the weights of training data D, the algorithm computes the misclassification rate, the sum of the weights of each of the training example t_i □ D that *were* misclassified. That is,

$$error(M_i) = \sum_{i}^{d} w_i * err(t_i); \tag{5}$$

Where $err\,(t_i)$ is the misclassification error of example t_i.. If the example t_i was misclassified, then is $err\,(t_i)$ 1. Otherwise, it is 0. The misclassification rate affects how the weights of the training examples are updated. If a training example was correctly classified, its weight is multiplied by $error\,(M_i)/(1-error(M_i))$. Once the weights of all of the correctly classified examples are updated, the weights for all examples including the misclassified examples are normalized so that their sum remains the same as it was before. To normalize a weight, the algorithm multiplies the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified examples are increased and the weights of correctly classified examples are decreased. Now the algorithm generates another new data set D_i from training data D with maximum weight values and continues the process until all the training examples are correctly classified. Or, we can set the number of rounds that the algorithm will iterate the process. To classify a new or unseen example use all the probabilities of each round (each round is considered as a classifier) and consider the class of new example with highest classifier's vote. The main procedure of proposed algorithm is described as follows:

Algorithm: An ensemble of classifiers using boosting and naïve Bayesian classifier.

Input: D, Training data D of labeled examples t_i.

Output: A classification model.

Procedure:

1. Initialize the weight w_i=1/n of each example t_i □ D, where n is the total number of training examples.
2. Generate a new dataset D_i with equal number of examples from D using selection with replacement technique.
3. Calculate the prior probability $P(C_j)$ for each class C_j in dataset D_i: $P(C_j) = \dfrac{\sum t_{i\to C_j}}{\sum_{i=1}^{n} t_i}$;
4. Calculate the class conditional probabilities $P(A_{ij}|C_j)$ for each attribute values in dataset D_i:

$$P(A_{ij}|C_j) = \frac{\sum_{i=1}^{n} A_{i\to C_j}}{\sum t_{i\to C_j}};$$

5. Classify each training example t_i in training data D with maximum posterior probabilities.

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

6. Updates the weights of each training examples t_i □ D, according to how they were classified. If an example was misclassified then its weight is increased, or if an example was correctly classified then its weight is decreased. To updates the weights of training examples the misclassification rate is calculated, the sum of the weights of each of the training example t_i □ D that *were* misclassified:

   $$error\ (M_i) = \sum_{i}^{d} w_i * err(t_i);$$

   Where *err* (t_i) is the misclassification error of example t_i. If the example t_i was misclassified, then is *err* (t_i) 1. Otherwise, it is 0. If a training example was correctly classified, its weight is multiplied by *error* $(M_i)/(1-error(M_i))$. Once the weights of all of the correctly classified examples are updated, the weights for all examples including the misclassified examples are normalized so that their sum remains the same as it was before. To normalize a weight, the algorithm multiplies the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified examples are increased and the weights of correctly classified examples are decreased.

7. Repeat steps 2 to 6 until all the training examples t_i □ D are correctly classified.

8. To classify a new/unseen example use all the probability set in each round (each round is considered as a classifier) and considers the class of new example with highest classifier's vote.

4. EXPERIMENTAL ANALYSIS

The performance of intrusion detection systems (IDS) are estimated by detection rates (DR) and false positives (FP). 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.

$$\text{DR} = \frac{Total\_detected\_attacks}{Total\_attacks} * 100 \qquad (6)$$

FP is defined as the total number of normal instances.

$$\text{FP} = \frac{Total\_misclassified\_process}{Total\_normal\_process} * 100 \qquad (7)$$

4.1 KDD99 Intrusion Detection Dataset

The access of intrusion detection dataset is strictly limited and cannot be shared in public domain, because the network data generated by IDS contain information about network topology, hosts and other confidential information's. The KDD 1999 cup benchmark intrusion detection dataset was used in the 3rd International Knowledge Discovery and Data Mining Tools Competition to evaluate the performance of various intrusion detection methods [36]. In 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. It 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. The KDD99 dataset contest uses a version of DARPA98 dataset. In KDD99 dataset, each example represents attribute values of a class in the network data flow, and each class is labeled either normal or attack.

The classes in KDD99 dataset can be categorized into five main classes: one normal class and four attack classes: probe, DOS, U2R, and R2L. Normal connections are the daily normal user behaviors. Denial of Service (DoS) attack causes the computing power or memory of a victim machine too busy or too full to handle legitimate requests. Remote to User (R2L) is an attack that a remote user gains access of a local user by sending packets to a machine over a network communication. User to Root (U2R) is an attack that an intruder begins with the access of a normal user account and then becomes a root-user by exploiting various vulnerabilities of the system. Probing (Probe) is an attack that scans a network to gather information or find known vulnerabilities. These four attacks are divided into 22 different attacks.

There are total 41 attributes in KDD99 dataset for each network connection that have either discrete or continuous values and divided into three groups. The first group of attributes is the basic features of network connection, which include the duration, prototype, service, number of bytes from source IP addresses or from destination IP addresses, and some flags in TCP connections. The second group of attributes in KDD99 is composed of the content features of network connections and the third group is composed of the statistical features that are computed either by a time window or a window of certain kind of connections. Table 1 show the number of examples in 10% training and testing data of KDD99 dataset.

Table 1. Number of examples in KDD99 dataset

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

4.2 Experimental Results

The experiments were performed by using an Intel Core 2 Duo Processor 2.0 GHz processor (2 MB Cache, 800 MHz FSB) with 1 GB of RAM. We tested the intrusion detection performance of the proposed learning algorithm with k-Nearest-Neighbor classifier (kNN), Decision Tree classifier (C4.5), Support Vector Machines (SVM), Neural Network (NN), and Genetic Algorithm (GA) by employing on the KDD99 benchmark intrusion detection dataset that is tabulated in Table 2 [37]-[40].

Table 2. Comparison of the results for the intrusion detection problem (Detection Rate %)

Method	Normal	Probe	DoS	U2R	R2L
Proposed Algorithm	100	99.95	99.92	99.55	99.60
kNN	99.60	75.00	97.30	35.00	0.60
C4.5	98.49	94.82	97.51	49.25	91.26
SVM	99.40	89.2	94.7	71.40	87.20
NN	99.60	92.7	97.50	48.00	98.00
GA	99.30	98.46	99.57	99.22	98.54

It has been successfully tested that effective attributes selection improves the detection rates for different types of network intrusions in intrusion detection. The performance of proposed algorithm on 12 attributes in KDD99 dataset is listed in Table 3.

Table 3. Result on reduce KDD99 dataset

Attack Types	DR (%)	FP (%)
Normal	100	0.03
Probing	99.95	0.36
Dos	100	0.03
U2R	99.67	0.10
R2L	99.58	6.71

5. CONCLUSIONS & FUTURE WORKS

In this paper, we introduce a new algorithm for adaptive intrusion detection based on boosting and naïve Bayesian classifier, which is an ensemble approach of boosting for improving the detection rates with low false positives in intrusion detection. The main propose of this paper is to improve the performance of naïve Bayesian classifier in intrusion detection. The naïve Bayesian classifier is popular data mining algorithm for classification problem that has several advantages such as it is easy to use and only one scan of training data is required. It can also easily handle the missing values by simply omitting the probability when calculating the likelihoods of membership in each class. We tested the performance of proposed algorithm with existing data mining algorithms and the experimental results manifest that the proposed algorithm achieved high detection rates and reduced the percentage of false positives for different types of network intrusions. The future works focus on applying other mining algorithms with this boosting approach for improving the detection rates in intrusion detection and also apply this algorithm in real world problem domain of intrusion detection.

6. ACKNOWLEDGMENTS

Support for this research received from Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh, and Department of Computer Science and Engineering, United International University, Bangladesh.

7. REFERENCES

[1] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Computer & Security, Vol. 28, 2009, pp. 18-28.

[2] Animesh Patcha, and Jugn-Min Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," Computer Networks, Vol. 51, 2007, pp. 3448-3470.

[3] Dan Zhu, G. Premkumar, Xiaoning Zhang, Chao-Hsien Chu, "Data Mining for Network Intrusion Detection: A Comparison of Alternative Methods," Decision Sciences, Vol. 32, No. 4, Fall 2001, pp. 635-660.

[4] Su-Yun Wu, and Ester Yen, "Data mining-based intrusion detectors," Expert Systems with Application`s, Vol. 36, Issue 3, Part 1, April 2009, pp. 5605-5612.

[5] Barbara, Daniel, Couto, Julia, Jajodia, Sushil, Popyack, Leonard, Wu, and Ningning, "ADAM: Detecting intrusion by data mining," IEEE Workshop on Information Assurance and Security, West Point, New York, June 5-6, 2001.

[6] Lee W., "A data mining and CIDF based approach for detecting novel and distributed intrusions," Recent Advances in Intrusion Detection, 3rd International Workshop, RAID 2000, Toulouse, France, October 2-4, 2000, Proc. Lecture Notes in Computer Science 1907 Springer, 2000, pp. 49-65.

[7] Lee W., Stolfo S., and Mok K., "Adaptive Intrusion Detection: A Data Mining Approach," Artificial Intelligence Review, 14(6), December 2000, pp. 533-567.

[8] Dewan Md. Farid, Nouria Harbi, and Mohammad Zahidur Rahman, "Combining Naïve Bayes and Decision Tree for Adaptive Intrusion Detection," International Journal of Network Security & Its Applications, Vol. 2, No. 2, April 2010, pp. 12-25.

[9] Dewan Md. Farid, Jerome Darmont, and Mohammad Zahidur Rahman, "Attribute Weighting with Adaptive NBTree for Reducing False Positives in Intrusion Detection," International Journal of Computer Science and Information Security, Vol. 8, No. 1, April 2010, pp. 19-26.

[10] Dewan Md. Farid, and Mohammad Zahidur Rahman, "Anomaly Network Intrusion Detection Based on Improved Self Adaptive Bayesian Algorithm," Journal of Computers, Academy Publisher, Vol. 5, No. 1, January 2010, pp. 23-31.

[11] Dewan Md. Farid, Nouria Harbi, Suman Ahmmed, Mohammad Zahidur Rahman, and Chowdhury Mofizur Rahman, "Mining Network Data for Intrusion Detection through Naïve Bayesian with Clustering," In Proc. of the International Conference on Computer, Electrical, System Science, and Engineering (ICCESSE 2010), June 28-30, 2010, Paris, France, pp. 836-840.

[12] Dewan Md. Farid, Nguyen Huu Hoa, Jerome Darmont, Nouria Harbi, and Mohammad Zahidur Rahman, "Scaling up Detection Rates and Reducing False Positives in

Intrusion Detection using NBTree," In Proc. of the International Conference on Data Mining and Knowledge Engineering (ICDMKE 2010), April 28-30, 2010, Rome, Italy, pp. 186-190.

[13] Dewan Md. Farid, Nouria Harbi, Emna Bahri, Mohammad Zahidur Rahman and Chowdhury Mofizur Rahman, "Attacks Classification in Adaptive Intrusion Detection using Decision Tree," In Proc. of the International Conference on Computer Science (ICCS 2010), 29-31 March, 2010, Rio De Janeiro, Brazil, pp. 86-90.

[14] Dewan Md. Farid, Jerome Darmont, Nouria Harbi, Nguyen Huu Hoa, and Mohammad Zahidur Rahman, "Adaptive Network Intrusion Detection Learning: Attribute Selection and Classification," In Proc. of the International Conference on Computer Systems Engineering (ICCSE 2009), December 25-27, 2009, Bangkok, Thailand, pp. 82-86.

[15] Dewan Md. Farid, and Mohammad Zahidur Rahman, "Anomaly Detection Model for Network Intrusion Detection using Conditional Probabilities," In Proc. of the 6th International Conference on Information Technology in Asia 2009 (CITA'09), 6th – 9th July 2009, Kuching, Sarawak, Malaysia, pp. 104-110.

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

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

[18] T. Pietraszek, and C. V. Berghe, "Defending against injection attacks through context-sensitive string evaluation," In Recent Advances in Intrusion Detection (RAID2005), Seattle, WA, Springer-Verlag, vol. 3858 of Lecture Notes in Computer Science, 2005, pp. 124–145.

[19] "The php group, php hypertext preprocessor," 2001-2004, web page at http://www.php.net

[20] "The phpbb group, phpbb.com," 2001-2004, web page at http://www.phpbb,com

[21] X. Xu, and X.N. Wang, "Adaptive network intrusion detection method based on PCA and support vector machines," Lecture Notes in Artificial Intelligence (ADMA 2005), LNAI 3584, 2005, pp. 696-703.

[22] Martin Roesch, "SNORT: The open source network intrusion system," Official web page of Snort at http://www.snort.org/

[23] L. C. Wuu, C. H. Hung, and S. F. Chen, "Building intrusion pattern miner for sonrt network intrusion detection system," Journal of Systems and Software, vol. 80, Issue 10, 2007, pp. 1699-1715.

[24] James P. Anderson, "Computer security threat monitoring and surveillance," Technical Report 98-17, James P. Anderson Co., Fort Washington, Pennsylvania, USA, April 1980.

[25] Dorothy E. Denning, "An intrusion detection model," IEEE Transaction on Software Engineering, SE-13(2), 1987, pp. 222-232.

[26] S.E. Smaha, and Haystack, "An intrusion detection system," in Proc. of the IEEE Fourth Aerospace Computer Security Applications Conference, Orlando, FL, 1988, pp. 37-44.

[27] W. Fan, W. Lee, M. Miller, S. J. Stolfo, and P. K. Chan, "Using artificial anomalies to detect unknown and known network intrusions," Knowledge and Information Systems, 2005, pp. 507-527.

[28] Y. Bouzida, and F. Cuppens, "Detecting known and novel network intrusions," Security and Privacy in Dynamic Environments, 2006, pp. 258-270.

[29] S. Peddabachigari, A. Abraham, and J. Thomas, "Intrusion detection systems using decision tress and support vector machines," International Journal of Applied Science and Computations, 2004.

[30] D. Barbara, N. Wu, and Suchil Jajodia, "Detecting novel network intrusions using Bayes estimators," In Proc. of the 1st SIAM Conference on Data Mining, April 2001.

[31] D. Barbara, J. Couto, S. Jajodia, and N. Wu, "ADAM: A tested for exploring the use of data mining in intrusion detection," Special Interest Group on Management of Data (SIGMOD), Vol. 30 (4), 2001.

[32] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naïve Bayes vs. decision trees in intrusion detection systems," In Proc. of the 2004 ACM Symposium on Applied Computing, New York, 2004, pp. 420-424.

[33] M. Panda, and M. R. Patra, "Network intrusion deteciton using naïve Bayes," International Journal of Computer Science and Network Security (IJCSNS), Vol. 7, No. 12, December 2007, pp. 258-263.

[34] M. Panda, and M. R. Patra, "Semi-naïve Bayesian method for network intrusion detection system," In Proc. of the 16th International Conference on Neural Information Processing, December 2009.

[35] Y. Freund, and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, Vol. 55, 1997, pp. 119-139.

[36] The KDD Archive. KDD99 cup dataset, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[37] Mukkamala S, Sung AH, and Abraham A, "Intrusion detection using an ensemble of intelligent paradigms," Journal of Network and Computer Applications, 2005, Vol. 2, No. 8, pp. 167-182.

[38] Chebrolu S, Abraham A, and Thomas JP, "Feature deduction and ensemble design of intrusion detection systems." Computer & Security, 2004, Vol. 24, No. 4, pp. 295-307.

[39] C. Elkan, 2007, Result of the KDD'99 Knowledge Discovery Contest [Online], Available: http://www-cse.ucsd.edu/users/elkan/clresults.html

[40] A. D. Joshi, "Applying the wrapper approach for auto discovery of under-sampling and over-sampling percentages on skewed datasets," M.Sc. Thesis, University South Florida, Tampa, 2004, pp. 1-77 [Online], Available: http://etd.fcla.edu/SF/SFE0000491/Thesis-AjayJoshi.pdf

8. AUTHORS PROFILE

Dewan Md. Farid is a doctoral candidate in the Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh. He obtained 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 a part-time faculty member in the Department of Computer Science and Engineering, United International University, Bangladesh. He is a member of IEEE and IEEE Computer Society. He has published 4 international journals and 9 international conference papers in the field of data mining, machine learning, and intrusion detection. He has participated and presented his papers in international conferences at France, Italy, Portugal, and Malaysia. He worked as a visiting researcher at ERIC Laboratory, University Lumière Lyon 2 – France from 01-09-2009 to 30-06-2010.

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.

Professor Dr. Chowdhury Mofizur Rahman had his B.Sc. (EEE) and M.Sc. (CSE) from Bangladesh University of Engineering and Technology (BUET) in 1989 and 1992 respectively. He earned his Ph.D from Tokyo Institute of Technology in 1996 under the auspices of Japanese Government scholarship. Prof Chowdhury is presently working as the Pro Vice Chancellor and acting treasurer of United International University (UIU), Dhaka, Bangladesh. He is also one of the founder trustees of UIU. Before joining UIU he worked as the head of Computer Science & Engineering department of Bangladesh University of Engineering & Technology which is the number one technical public university in Bangladesh. His research area covers Data Mining, Machine Learning, AI and Pattern Recognition. He is active in research activities and published around 100 technical papers in international journals and conferences. He was the Editor of IEB journal and worked as the moderator of NCC accredited centers in Bangladesh. He worked as the organizing chair and program committee member of a number of international conferences held in Bangladesh and abroad. At present he is acting as the coordinator from Bangladesh for EU sponsored eLINK project. Prof Chowdhury has been working as the external expert member for Computer Science departments of a number of renowned public and private universities in Bangladesh. He is actively contributing towards the national goal of converting the country towards Digital Bangladesh.