Advances in Science, Technology and Engineering Systems Journal Vol. 2, No. 3, 812-818 (2017)

www.astesj.com

Special Issue on Recent Advances in Engineering Systems

ASTESJ
ISSN: 2415-6698

Network Intrusion Detection System using Apache Storm

Muhammad Asif Manzoor[*], Yasser Morgan

Faculty of Engineering and Applied Sciences, University of Regina, SK, Canada

A R T I C L E I N F O

Article history:
Received: 05 April, 2017
Accepted: 27 May, 2017
Online: 19 June, 2017

Keywords:
Network Intrusion Detection
Support Vector Machine
Apache Storm
LibSVM
KDD 99

A B S T R A C T

Network security implements various strategies for the identification and prevention of security breaches. Network intrusion detection is a critical component of network management for security, quality of service and other purposes. These systems allow early detection of network intrusion and malicious activities; so that the Network Security infrastructure can react to mitigate these threats. Various systems are proposed to enhance the network security. We are proposing to use anomaly based network intrusion detection system in this work. Anomaly based intrusion detection system can identify the new network threats. We also propose to use Real-time Big Data Stream Processing Framework, Apache Storm, for the implementation of network intrusion detection system. Apache Storm can help to manage the network traffic which is generated at enormous speed and size and the network traffic speed and size is constantly increasing. We have used Support Vector Machine in this work. We use Knowledge Discovery and Data Mining 1999 (KDD'99) dataset to test and evaluate our proposed solution.

1. Introduction

Network security is of increasing importance than ever with increased usage of network-based computing resources. Several network security mechanisms like encryption, firewalls, cryptography, access control, authentication, and intrusion detection are used to provide network security. These techniques are used to detect and prevent the malicious activities and network attacks. Network intrusion detection is an essential task for any network based computing environment. Network Intrusion Detection system continuously monitors the network traffic and perform in-depth packet analysis for malicious activities and attacks in real time. Once the intrusions are detected, appropriate actions can be taken to mitigate the threat against the computing and network resources. Currently, computer networks are generating a huge amount of data traffic at enormous speed and this data generation speed is continuously increasing. Network traffic also contains different types (variety) of data. Network traffic satisfies all the three V's of big data; volume, velocity, and variety. The focus of this paper is to propose a network intrusion detection mechanism that can deal with all the 3 V's of big data.

This paper is an extension of work originally presented in 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) [1]. In original work, we have used anomaly-based approach for network intrusion detection. Anomaly based approaches use machine learning algorithms to identify the abnormal behavior from the normal data traffic. We have used statistical approaches for feature reduction and support vector machine with the linear kernel for the classification process. We also have proposed an Apache Storm topology for the real-time big data streaming application. The current work uses Radial Basis Function (RBF) kernel for the support vector machine. This paper discusses the class imbalance problem and its possible solutions. We also discuss multiple techniques to reduce the false positive and false negative rates in the context of network intrusion detection system.

Mostly, network intrusion detection system follows one of the two major detection mechanism; Anomaly-based network intrusion detection and Signature based network intrusion detection. Some researchers also have proposed hybrid approaches. Each detection approach has its own strengths and weaknesses.

1.1. Signature Based Network Intrusion Detection

Signature based or misuse intrusion detection systems have better detection rate as compared to anomaly based systems in case of known network attacks. This type of intrusion detection systems scans predetermined signatures in network traffic packets and recognizes predefined patterns of network intrusions and attacks.

[*]Muhammad Asif Manzoor, Faculty of Engineering and Applied Sciences, University of Regina, SK, Canada, AsifManzoor@uregina.ca

The signature based system cannot detect new potential threats until their signature is added to the database. During the time, when a new threat is detected and its signature is applied, these systems cannot detect the new threats. Malicious strings in TCP/IP packets, abnormal flag combination, request for non-existing services, and intrusion attack signature are few of the approaches [2], [3] used to design signature based network intrusion detection systems.

1.2. Anomaly Based Network Intrusion Detection

Anomaly based system try to capture the behavior of incoming network traffic. These systems establish a baseline model for normal user/network behavior against the abnormal behavior. This model is created by analyzing the network traffic over a period of time and train the system to classify the normal traffic and malicious traffic. The baseline model reflects the valid activities of applications, hosts, users, and network. All the incoming network traffic is scanned using the defined model to classify it into normal and intrusion network packet. Alarms are raised for all the packets which are classified as intrusions. Anomaly based detection systems are good to detect new network threat at the cost of increased false alarm rate (both false positive and false negative detection). Despite this problem, anomaly based detection is an indispensable method due to increase in new type intrusion attacks.

The rest of paper is organized as follow, anomaly based intrusion detection systems are review in section II. Section III provides the detailed description of KDD 99 data set and also discusses the pre-processing techniques used in this work. Implementation details and methodology is discussed in Section IV. Experimental results are discussed in section V. Class imbalance problem and false positive/negative problem for network intrusion detection are discussed in Section VI and the paper is concluded in Section VII.

2. Literature Review

Machine learning algorithms are used to design anomaly based intrusion detection system. Both unsupervised and supervised learning methods are used in this domain. Although many intrusion detection methods are proposed using various different machine algorithms we will review Artificial Neural Network (supervised learning) and K-Mean Nearest neighbor (unsupervised learning) based approaches.

Artificial Neural Networks are used by many researchers as supervised learning algorithm to train the intrusion detection technique. Rapake et al. [5] proposed to use numbers of system calls that are executed on the host machine. These numbers are utilized for neural network training. Han and Cho [6] proposed to use evolutionary neural networks to detect anomalies based on learning the behavior of the program. Authors have used 1999 DARPA IDEVAL data set to evaluate their proposed method. Liang et al. [4] proposed to apply Fisher feature selection algorithm on KDD 99 dataset for performance enhancement. Authors used artificial neural networks to develop their intrusion detection system.

K-Mean Nearest Neighbor (KNN) is a clustering algorithm based on unsupervised learning method. Wang et al. [7] used KDD 99 dataset and defined new features based on density, cluster centers, and nearest neighbors. KNN is used to classify the normal traffic and network intrusion. Li et al. in [8] combined KNN algorithm with Particle Swarm Optimization (PSO) to design hybrid system. The K-mean algorithm suffers from premature convergence. PSO algorithm helps to avoid premature convergence in the proposed method. Xian et al. [9] proposed to unite fuzzy KNN algorithm with clonal selection algorithm to design network intrusion detection. Jiang et al. [10] used incremental KNN algorithm to detect intrusions. Authors used outlier factor to calculate the deviation degree of the cluster in this work.

Multiple network intrusion detection systems are proposed using data mining algorithms. Han et al. [11] used data mining approach to detect intrusions. They have analyzed attributes of network traffic protocol to identify misuse signature. They also have analyzed network packet contents for intrusion signatures. Qin and Hwang [12] proposed internet trace technique which discards few non-functionary frequent episodes rules dynamically. These episode rules are utilized to recognize abnormal sequences in network traffic connections. Otey et al. [13] develop a general purpose tunable algorithm to detect outlier/anomalies. The proposed algorithm can work with mixed attribute dataset as well dynamic and streaming data sets. KDD 99 intrusion dataset is one of the dataset used to test and evaluate the proposed tunable algorithm.

Gondal et al. [14][14] determined center of clusters on basis of diversity. These Diversity-based centroids are used to develop network intrusion detection system. Xiao et al. in [15] used Bayesian Network Model Averaging in their proposed work and compared it with Naïve Bayes classifier and Bayesian Network classifier. Authors have evaluated their work over NSL-KDD dataset.

Jeong et al. [16] proposed in their work to used Discrete Wavelet Transform to extract the features from network traffic. Principal Component Analysis (PCA) is applied to the extracted features to identify principal components and a visual analytics tool is proposed for intrusion detection. Le et al. [17] used deep learning approach in their network intrusion detection system. Authors explored various optimizers with Long Short-Term Memory Recurrent Neural Network.

Table I: Details of Dataset

	Training	**Testing**
Normal	97280	60623
Probe	4107	4166
DoS	391458	229853
U2R	52	13939
R2L	1124	2478
Total	494021	321026

Table II: Network Attacks and Categories

Category	Network Attacks
Denial of Service (DoS)	back, land, neptune, pod, smurf, teardrop, **apache2, mailbomb, processtable**
Probe	ipsweep, nmap, portsweep, satan, **mscan, saint**
User to Root (U2R)	ftpwrite, guespasswd, imap, multihop, phf, spy, warezclient, Warezmaster, **httptunnel, named, sendmail, snmpgetattack, xlock, xsnoop**
Remote to Local (R2L)	bufferiverflow, loadmodule, perl, rootkit, **ps, snmpguess, sqlattack, worm, xterm**

3. Knowledge and Discovery 1999 Dataset (KDD 99)

Knowledge and Discovery 1999 Network Intrusion Detection dataset (KDD 99) [18] is used in this work to test and evaluate the performance of the proposed method. KDD 99 is publicly available dataset and widely used by various researchers. The complete KDD 99 dataset contains around five million records for training and 3 million records for testing tasks. Due to the large size of the original dataset, smaller version of KDD 99 dataset is frequently used for evaluating intrusion detection systems. This smaller version contains 10% of testing and training records. We also have tested our intrusion detection system using smaller dataset. KDD 99 dataset contains normal records as well as malicious records. Twenty two types of attacks are included in training dataset along with normal network packets. Seventeen additional attacks are added to the testing dataset and can be used to determine whether the proposed algorithm can detect new attacks or not. Network attacks are divided into four categories; Denial of Service (DoS), Probe, User-to-Root (U2R), and Remote-to-Local (R2L). Table I provides the distribution of training and testing dataset into normal and intrusion packets. KDD 99 dataset consists of 494021 training records and 311029 testing records. The division of network attacks into categories is given in Table II. Network attacks in bold are new attacks added to testing dataset only.

Each record has 41 attributes; description of these attributes is given below:

- 9 basic and single connection derived (SCD) header features,
- 9 time based multiple connection derived (MCD) header features,
- 10 host based multiple connection derived (MCD) header features, and
- 13 content based features collected from traffic payloads.

3.1. Pre-Processing

KDD 99 data set is pre-processed in order to make it appropriate for the machine learning algorithm. The second reason for the pre-processing is to enhance the intrusion detection rate. Pre-processing is performed in three steps.

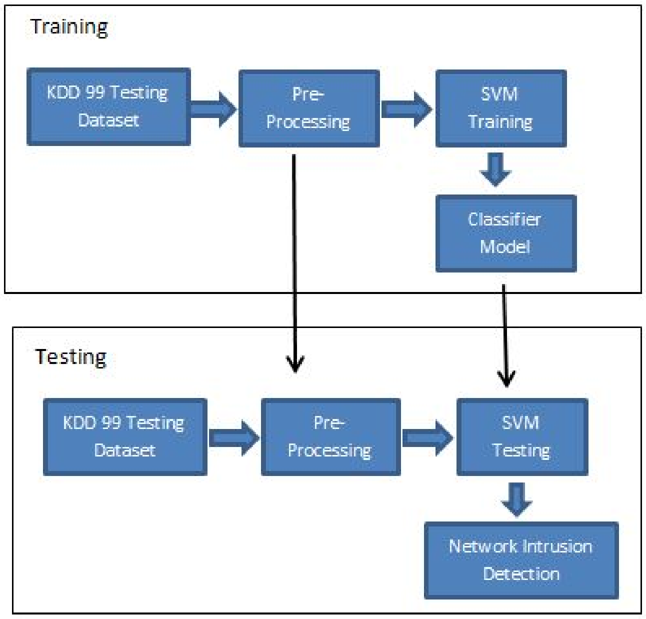

Figure 1: Network Intrusion Detection System Architecture

1. Each record in the dataset consists of numeric as well as categorical attributes. Textual data is used for categorical attributes. Support vector machine algorithm requires numeric data (either discrete or continuous). The first step in pre-processing is to convert this categorical attributes to numeric attributes. The dataset contains three categorical attributes while rest of the thirty eight attributes are numeric. Every category of an attribute is assigned a specific number.

2. Next, we have normalized all the numeric attributes in the training dataset. Testing dataset attributes are normalized using ratios determined for training dataset, so both (training and testing) subsets are normalized with same ratios. Normalization process converts all attributes to the same range; hence each attribute has a similar effect during the training process. The normalization process also significantly reduces the training time for support vector machine algorithm.

3. We applied statistical operations on the training dataset to reduce the features. Same features are dropped from testing dataset. These dropped attributes had a negative impact on the performance of network intrusion detection system. Reduce the number of attributes have a positive effect on the computation time.

4. We have used LibSVM library for SVM implementation; it requires training and testing data in a specific format. Lastly, we have converted the dataset to LibSVM compatible format.

4. Methodology

The architecture of Support Vector based Network Intrusion Detection system is given in Figure 1. KDD 99 training data set is pre-processed and Support Vector Machine (SVM) algorithm is applied to build a model. Pre-processing information determined for the training dataset is used to pre-process the training dataset. SVM model built during the training process is used to predict the category during the testing process.

4.1. Support Vector Machine

Support Vector Machine algorithm [19] is developed to solve binary classification problems. SVM is supervised learning algorithm which can be used for classification and regression problems and it is based on statistical learning approach. The fundamental idea of SVM is to identify hyperplanes (decision boundaries) between two categories based on the training data and one providing the maximum separation between two categories is selected as shown in Figure 2. This hyperplane provides the optimal separation between two categories. The identification of optimal hyperplane is considered as an optimization problem. Testing examples are projected and the category is predicted based on their position with respect to the optimal hyperplane. Cover's theorem [20] states; if linearly non-separable dataset is transformed using non-linear kernel function then it is probable that the transformation process will generate linearly separable dataset in higher dimensional space. Support vector machine also uses different types of non-linear kernel function to create linearly separable dataset and increase classification performance.

Let we have a training dataset S= $\{(x_1, y_1), (x_2, y_2),\ldots, (x_n, y_n)\}$, where $x_i \in R^n$ and it represents input feature vector and $y_i \in \{-1, 1\}$ is the category. Let weight and bias of hyperplane is given by w and b. The non-linear Kernel function is applied on training dataset to transform it into linearly separable categories and $\varphi(x)$ is used to represent it. The hyperplane between the two categories can be defined as:

$$w.\varphi(x) + b = 0 \qquad (1)$$

The optimization problem for calculation of w and b is:

Minimize

$$\varphi(w) = ½\, \|w\|^2 \qquad (2)$$

Subject to

$$y_i(w.\varphi(x) + b) \geq 1$$

New variables ζ_i (slack variable) and C (regularization constant) are included in above optimization problem:

Minimize

$$\varphi(w,\zeta) = ½\, \|w\|^2 + C \sum_{i=1}^{N} \zeta_i \qquad (3)$$

Subject to

$$y_i(w.\varphi(x) + b) \geq 1 - \zeta_i,\ \zeta_i \geq 0$$

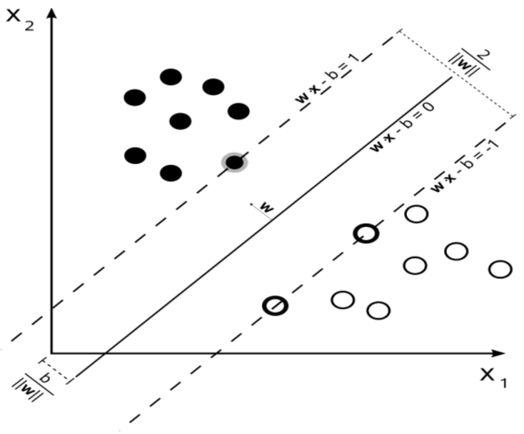

Figure 2: Decision Boundary with maximum margin

Where ζ is used to relax the hard margin constraint and C is used to manage the trade-off between classification error and maximal margin of separation.

Multiclass SVM is often constructed using basic binary SVMs to deal with real world problems like pedestrian detection [21], medical diagnosis [22], sentiment analysis [23] and many others. For multiclass SVM, 1-vs-1 [24] and 1-vs-all [25] are two approaches used to build basic binary SVMs.

We have used LibSVM [26] Java API in this work for SVM implementation. LibSVM provides both regression and classification implementations. We have used C-SVM approach to classifying the network traffic data into normal and intrusions. Radial Basis Function (non-linear kernel function) is used in this work for transformation into higher dimensional space. The value of gamma used here is 1 and the value of C is 6. As kernel implementation is part of LibSVM library; hence we are considering it a part of SVM training process and not a part of pre-processing. Finally, we have used the 1-vs-1 multiclass approach in this work.

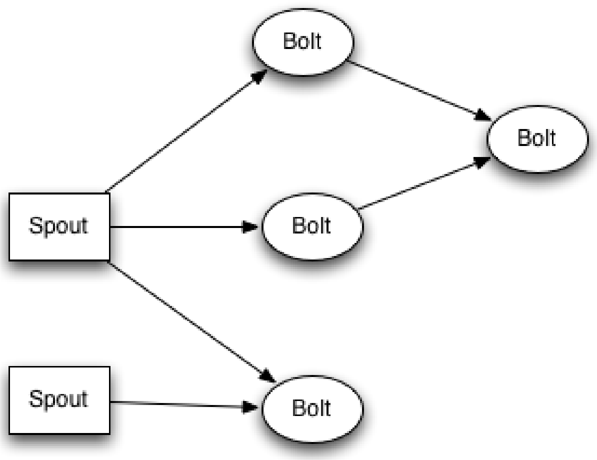

Figure 3: Apache Storm Topology

4.2. Apache Storm

Apache Storm is a scalable distributed framework that can be easily configured on cloud services or clusters. It is developed to process real-time big streaming data. Spouts and bolts are the basic building blocks for Apache Storm applications. Spouts are used as data sources; data is forwarded in form of tuples to the bolts. Bolts are the computation unit of the Apache Storm. Bolts process and evaluate the incoming tuples and can forward the results in form of tuple to other bolts for further processing. Spouts and bolts are combined to form a topology. Topologies are directed acyclic graphs and used to develop Apache Storm applications. An Apache Storm topology is shown in Figure 3. The edges of the topology show the flow of data. Bolts and spouts are vertices in the topology. We can create many instances of every bolt and spout to speed up the processing.

Apache Storm framework offers distributed and fault tolerant computing. An Apache Storm cluster is shown in Figure 4. The cluster consists of one or multiple worker nodes and only one master node. Worker nodes are named as supervisors while the master node is named as nimbus. Apache Storm also requires ZooKeeper cluster to provide coordination between the master node and worker node(s).

The proposed Apache Storm topology is given in Figure 5. This topology has one spout and three bolts. The detail of the bolts and spout is given below:

1. **Input Reader (Spout)** is the only data source in this topology. KDD 99 dataset is stored in a text file. This spout reads the KDD 99 dataset and each record is forwarded as a tuple to the next bolt.
2. **Data Pre-Processer (Bolt)** receives the tuples from Input Reader and performs pre-processing as explained in Section 3. It converts the categorical data to numeric data, perform normalization and feature reduction. Finally, it converts the data into LibSVM compatible format.

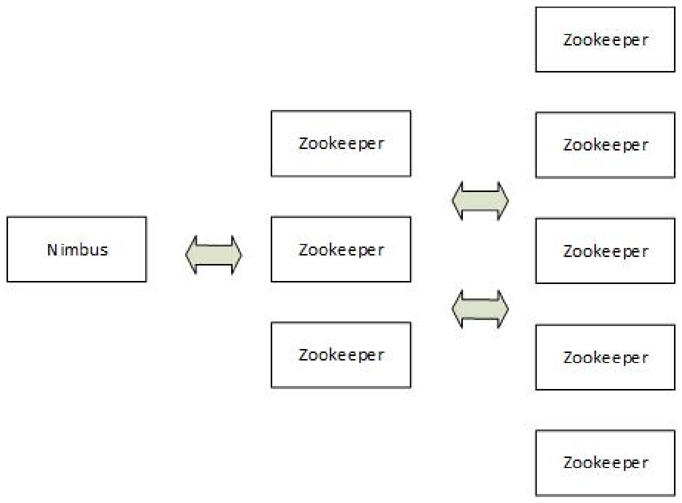

Figure 4: Apache Storm Cluster

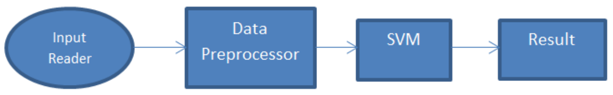

Figure 5: Apache Storm Topology

3. **Support Vector Machine (Bolt)** performs the classification process for the network intrusion detection system.
4. **Result Aggregator (Bolt)** is the last processing unit in this topology. It aggregates the classification results. It stores these results on a text file for further analysis.

5. Experimental Results and Discussion

5.1. Experimental Setup

Apache Storm 0.9.6 is configured on Ubuntu 12.04. All the experiments are performed on 3.4 GHz Intel Core i7 processor with 16.00 GB of memory.

5.2. Evaluation and Analysis

We have used KDD 99 intrusion detection dataset to test our SVM based method. The testing dataset contains 18729 records belonging to new network intrusion attacks (17 new network attacks included) which are not included in the training dataset. Whereas 292300 records belong to network intrusion attacks which are also part of the training dataset. The accuracy for intrusion detection of the proposed method is 98.03% when only known attacks are considered (Network attacks used during training process). This intrusion detection rate drops to 92. 60% when all the network attacks are considered during the testing process. All the discussion, following this point, will be based on the complete testing dataset. The classification results for complete dataset (new network attacks included) of the proposed intrusion detection method are given in Table III. A simplified confusion matrix is given in Table IV; which only divides the classification results into normal and malicious network traffic.

Table III: Confusion Matrix for Network Intrusion Detection System

	Normal	Dos	Probe	R2L	U2R
Normal	60294	66	225	7	1
DoS	6571	223266	16	0	0
Probe	760	441	2965	0	0
R2L	12849	91	72	927	0
U2R	2462	3	0	7	6

Table IV: Simplified Confusion Matrix

	Normal	Attack
Normal	60294	299
Attack	22642	227794

The performance of the can also be determined using False Positive Rate (FPR), False Negative Rate (FNR), Precision and Recall. These four performance evaluators are calculated below using simplified confusion matrix (Table IV).

FPR = FP / (FP + TN) = 22642 / (22642 + 227794) = 0.09

FNR = FN / (FN + TP) = 299 / (299 + 60294) = 0.005

Precision = TP / (TP+FP) = 60294 / (60294+22642) = 0.73

Recall = TP / (TP+FN) = 60294 / (60294 + 299) = 0.995

For a good classification rate, FPR and FNR must be close to zero while Precision and Recall must be close to one. Other than Precision, all other performance parameters indicate good classification performance.

Normal traffic and DoS attacks are recognized correctly mostly. Whereas Remote to Local (R2L) and User to Root (U2R) are the two categories which are recognized incorrectly most of the time. Class imbalance is the main reason behind this shortcoming. R2L and U2R have the least number of training samples. Similarly, probe attack category also has relatively less training examples. While remaining two categories have the majority of the training examples; around 98.9% of the dataset. This class imbalance creates biased results during the training process. We will discuss few techniques to solve class imbalance problem in the next section.

The proposed network intrusion detection system is supposed to handle real-time streaming data traffic. The training process is done offline and it is not a real-time task. This system can process roughly 13,600 packets in a second for testing/prediction purposes. This processing speed is achieved with a single general purpose computer. The processing speed can be further increased by using a cluster of dedicated servers. This processing speed is a good indicator for real-time big streaming data scenario.

Support Vector Machines are machine learning algorithm used for supervised classification tasks whereas Apache Storm is development platform used to develop real-time big streaming data processing applications. Apache Storm supports multiple languages for coding. We have used Java in our work. The proposed system achieves good classification results along with a good processing speed.

6. Performance Enhancement for Network Intrusion Detection Systems

Class imbalance refers to the uneven representation of classes in training dataset. The skewed distribution causes performance degradation for many machine learning algorithms [27]. The methods to solve class imbalance problem are generally divided into two categories; data level and algorithm level solutions.

6.1. Data Level Approach

Data level approaches apply different techniques to the dataset before training process. It can be considered as pre-processing step to accommodate the class distribution. Sampling, under-sampling, and over-sampling, is a data-level approach to managing class distribution. Over-sampling duplicates the records from the minority category whereas under-sampling removes records from the majority category to minimize the imbalance ratio between different categories in training data.

Yen et al. [28] proposed to use cluster based approach to solve the class imbalance problem by selecting representative data. Clusters are used to identify the representative data to achieve better prediction rate for minority categories. Yu et al. [29] used vector quantization to under-sample the majority class and constructed representative local models. Synthetic Minority Over-Sampling Technique (SMOTE) [30] is an adaptive data-level approach for over-sampling. It computes the probability distribution to model minor categories and use this probability distribution to add new examples to the smaller classes. Guo and Viktor [31] proposed to combine boosting and data generation to over-sample the minority classes. This approach is named as DataBoost-IM algorithm.

6.2. Algorithm Level Approach

In algorithm level approach, specialized machine learning algorithms are used to tackle class imbalance problem.

One of the techniques used is to develop modified version of the basic learning algorithm which can manage imbalanced classes. z-SVM [32] and GSVM-RU [33] are modified version of SVM to manage imbalanced class distribution. z- SVM tries to maximize g-mean value by moving hyperplane using a parameter z. GSVM-RU uses granular computing to enhance the classification accuracy.

Another algorithm level approach is to use cost sensitive learning; whenever a misclassification occurs, an expensive cost is imposed on the classifier. Cao et al. [34] proposed Particle Swarm Optimization (PSO) based cost sensitive artificial neural network for imbalanced data classification.

7. Conclusion

Network intrusion detection is an important component of network management and it is a defense mechanism for network security. Anomaly based intrusion detection detects anomalous behavior or new network attacks which signature-based intrusion detection cannot. Higher processing speed is required to handle incoming real-time data traffic. The proposed Network Intrusion detection system can process 13,600 packets in a second on a single general purpose computer. Apache Storm based network intrusion detection system can achieve higher processing by deploying a cluster of computers. Data pre-processing techniques are implemented to reduce the number of feature; which reduces the training and testing time and increased the classification accuracy.

In this paper, we outline the machine learning based method for anomaly based intrusion detection. We propose to utilize support vector machine as supervised learning algorithm to classify the incoming network traffic into normal and network intrusion attacks (DoS, Probe, R2L, U2R). KDD 99 intrusion detection dataset is used in this work. The data set is pre-processed in order to make incompatible with support vector machine algorithm as well as with LibSVM library. The experimental results show the feasibility of the proposed intrusion detection method. The performance can be further enhanced by solving class imbalance problem as discussed in section 6. This paper suggests using Support Vector Machine for classification process and Apache Storm as development platform to develop network intrusion detection system to handle big data characteristics of the network data traffic.

References

[1] Manzoor, Muhammad Asif, and Yasser Morgan. "Real-time Support Vector Machine based Network Intrusion Detection system using Apache Storm." Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual. IEEE, 2016.

[2] Bro, Bro Intrusion Detection System 2002.

[3] SNORT, SNORT: The open source network intrusion detection system 2002.

[4] Hu, Liang, et al. "An improved intrusion detection framework based on Artificial Neural Networks." Natural Computation (ICNC), 2015 11th International Conference on. IEEE, 2015.

[5] A. Rapaka, A. Novokhodko, and D. Wunsch, “Intrusion detection using radial basis function network on sequences of system calls,” in Proc. Int. Joint Conf. Neural Netw., 2003, vol. 3, pp. 1820–1825.

[6] S. J. Han and S. B. Cho, “Evolutionary neural networks for anomaly detection based on the behavior of a program,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 36, no. 3, pp. 559–570, Jun. 2006.

[7] Xiujuan Wang, Chenxi Zhang and Kangfeng Zheng, "Intrusion detection algorithm based on density, cluster centers, and nearest neighbors," in China Communications, vol. 13, no. 7, pp. 24-31, July 2016.

[8] Z. Li, Y. Li, and L. Xu, “Anomaly intrusion detection method based on k-means clustering algorithm with particle swarm optimization,” in Proceedings of the 2011 International Conference of Information Technology, Computer Engineering and Management Sciences. Washington, DC, USA: IEEE Computer Society, 2011, pp. 157–161.

[9] J. Xian, F. Lang, and X. Tang, “A novel intrusion detection method based on clonal selection clustering algorithm,” in Proc. Int. Conf. Mach. Learn. Cybern., 2005, vol. 6, pp. 3905–3910.

[10] S. Jiang, X. Song, H.Wang, J. Han, and Q. Li, “A clustering-based method for unsupervised intrusion detections,” Pattern Recognit. Lett., vol. 27, no. 7, pp. 802–810, May 2006.

[11] H. Han, X. L. Lu, and L. Y. Ren, “Using data mining to discover signatures in network-based intrusion detection,” in Proc. Int. Conf. Mach. Learn. Cybern., 2002, vol. 1, pp. 13–17.

[12] M. Qin and K. Hwang, “Frequent episode rules for Internet anomaly detection,” in Proc. IEEE Int. Symp. Netw. Comput. Appl., 2004, pp. 161–168.

[13] M. E. Otey, A. Ghoting, and S. Parthasarathy, “Fast distributed outlier detection in mixed-attribute data sets,” Data Min. Knowl. Discov., vol. 12, no. 2/3, pp. 203–228, May 2006.

[14] Gondal, M.S.; Malik, A.J.; Khan, F.A., "Network Intrusion Detection Using Diversity-Based Centroid Mechanism," in Information Technology - New Generations (ITNG), 2015 12th International Conference on , vol., no., pp.224-228, 13-15 April 2015

[15] Liyuan Xiao; Yetian Chen; Chang, C.K., "Bayesian Model Averaging of Bayesian Network Classifiers for Intrusion Detection," in Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International , vol., no., pp.128-133, 21-25 July 2014

[16] Jeong, Dong Hyun, Bong-Keun Jeong, and Soo-Yeon Ji. "Designing a hybrid approach with computational analysis and visual analytics to detect network intrusions." Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual. IEEE, 2017.

[17] T. T. H. Le, J. Kim and H. Kim, "An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization," 2017 International Conference on Platform Technology and Service (PlatCon), Busan, South Korea, 2017, pp. 1-6.

[18] The UCI KDD Archive University of California, Irvine, 1999: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[19] Cortes, C., Vapnik, V.: Support vector networks. Mach. Learn. 20(3), 273–297 (1995)

[20] Devroye, Luc, László Györfi, and Gábor Lugosi. A probabilistic theory of pattern recognition. Vol. 31. Springer Science & Business Media, 2013.

[21] J. Baek, J. Kim and E. Kim, "Fast and Efficient Pedestrian Detection via the Cascade Implementation of an Additive Kernel Support Vector Machine," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 902-916, April 2017.

[22] H. Ma, T. Tan, H. Zhou and T. Gao, "Support Vector Machine-recursive feature elimination for the diagnosis of Parkinson disease based on speech analysis," *2016 Seventh International Conference on Intelligent Control and Information Processing (ICICIP)*, Siem Reap, Cambodia, 2016, pp. 34-40.

[23] Ye Fei, "Simultaneous Support Vector selection and parameter optimization using Support Vector Machines for sentiment classification," *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2016, pp. 59-62.

[24] Kreßel, U.: Pairwise classification and support vector machines. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods: Support Vector Learning, pp. 255–268. MIT Press, Cambridge (1999)

[25] Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)

[26] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.

[27] Wang, Shuo, and Xin Yao. "Multiclass imbalance problems: Analysis and potential solutions." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.4 (2012): 1119-1130.

[28] Yen, Show-Jane, and Yue-Shi Lee. "Cluster-based under-sampling approaches for imbalanced data distributions." *Expert Systems with Applications* 36.3 (2009): 5718-5727.

[29] Yu, Ting, et al. "A hierarchical VQSVM for imbalanced data sets." *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. IEEE, 2007.

[30] Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." *Journal of artificial intelligence research* 16 (2002): 321-357.

[31] Guo, Hongyu, and Herna L. Viktor. "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach." *ACM Sigkdd Explorations Newsletter* 6.1 (2004): 30-39.

[32] Imam, Tasadduq, Kai Ting, and Joarder Kamruzzaman. "z-SVM: an SVM for improved classification of imbalanced data." *AI 2006: Advances in Artificial Intelligence* (2006): 264-273.

[33] Tang, Yuchun, et al. "SVMs modeling for highly imbalanced classification." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.1 (2009): 281-288.

[34] Cao, Peng, Dazhe Zhao, and Osmar R. Zaïane. "A PSO-based cost-sensitive neural network for imbalanced data classification." *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2013.