ISSN No. 0976-5697

Volume 4, No. 10, Sep-Oct 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Accuracy Improvement of IDS via Filter Parallelization

Maryam Amanifar
Department of Computer, Science and Research Branch,
Islamic Azad University,
Khouzestan, Iran
amanifar_1200@yahoo.com

DR. Mohammad Saniee Abadeh
Electrical and Computer Engineering Faculty
Tarbiat Modares University
Tehran, Iran
saniee@modares.ac.ir

Abstract: Daily increase in using the computer network and internet technology services from one hand and attacking the computer networks from other hand which has caused the Intrusion Detection Systems (IDS) become a very important part of the computer system immunity. It has also become an important field of research in the computer system networks. Until now, many different approaches to enhance the applicability of IDSs are presented such as Machine Learning and Data Mining. In this paper an approach for enhancing the accuracy in the IDSs is presented. In order to compare the applicability of this approach with earlier presented approaches, a set of similar data KDD99 have been used. SVM classifier and K-NN have been used to detect intrusion with high classification rate.

Keywords: IDS; K-NN; SVM; KDD99

I. INTRODUCTION

One of the big problems in recent years has had a dramatic effect in internet there is a intrusion. Network attacks have increased in number and severity over the past few years [3, 5]; therefore, intrusion detection systems (IDS) have become an essential component of computer security to supplement existing defenses [2]. Today intrusion detection has caught researchers' attention greater than ever. Its development and improvement have been set with the highest priority by academia, government, research institutes, and industrial corporations. It examines activities from computer users and identifies inappropriate, incorrect, or anomalous activities within computers or networks. Until now, many different approaches to enhance the applicability of IDSs are presented, such as data mining and machine learning. Hesham Altwaijry used a Bayesian classifier for detect intrusion (2012) [3]. And he used a Multi-Layer Bayesian Based Intrusion Detection System (2011) [5]. Chou used a fuzzy clustering technique for intrusion detection system(2007) [1]. Cheng (2008) in his paper proposes a multiple-level hybrid classifier, a novel intrusion detection system, which combines the supervised tree classifiers and unsupervised Bayesian clustering to detect intrusions [2]. Cemerlic used a Network Intrusion Detection Based on Bayesian Networks. Romero compared four machine learning techniques for spam filtering in blog comments. The machine learning techniques are: Naïve Bayes, K- nearest neighbors, neural networks and support vector machines [8].

KDD99 dataset

The concept of detecting abnormal behavior of computer users was first introduced by Anderson in 1980. In 1999, Lincoln Laboratory at MIT created the KDD99 data set, which is known as "DARPA Intrusion Detection Evaluation Data Set". To test our IDS system we used the DARPA KDD99 Intrusion Detection Evaluation dataset. It is made up of a large number of network traffic activities that include both normal and malicious connections. The KDD99 data set includes three independent sets; "whole KDD", "10% KDD", and "corrected KDD". In our experiments we have used the "10% KDD" and the "corrected KDD" as our training and testing set, respectively [4]. Table 1 summarizes the number of samples in each dataset:

Table I: KDD dataset

Dataset	Normal	DoS	Probing	U2R	R2L	Total
Whl DD	972,780	3,883,370	41,102	52	1126	4,898,430
10% KDD	97,278	391,458	4107	52	1126	494,020
KDD corr	60,593	229,853	4166	228	16,189	311,029

The training set contains a total of 22 training attack types. Additionally the "corrected KDD" testing set includes an additional 17 attack types. Therefore there are 39 attack types that are included in the testing set and these attacks can be classified into one of the four main classes;

DOS: Denial of Service attacks.
Probe: another attack type sometimes called Probing.
U2R: User to Root attacks.
R2L: Remote to Local attacks.

The KDD-99 network TCP connections have 41-features per connection (Records) these features can be divided into four categories [1, 6].
Basis features: Features 1-9 are the basic features that are derived from packet header without inspecting the payload.

© 2010, IJARCS All Rights Reserved

Content Features: Domain knowledge is used to assess the payload of the original TCP packets. This includes features such as the number of failed login attempts;
Time-based Traffic Features: These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval;
Host-based Traffic Features: Utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

A. Distance Algorithm

In this method must first obtain the center of each class. In training phase, center of normal data training and center of attacks data training are calculated by averaging the data in each class are determined. In the testing phase, we must be specified the class of each test data using the trained system. To determine the class of each test data, the Euclidean distance between the data center and the normal class to class Center obtained traces of intrusion. The data belongs to the class that has the least distance to the center of the classroom [9].

B. K-NN

The KNN classifier measures the distance between two data points P and Q by Euclidean distance (1), (2). The distance actually represents their similarity. The shorter the distance between them, the more similar they are [9].

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} \quad (1)$$

$$\sqrt{\sum_{i=1}^{n} (pi - qi)^2} \quad (2)$$

Where pi and qi are the values of the ith attribute of points P and Q respectively. The final similarity score of a data point being classified is the average of its Euclidean distances from the closest k normal points.

C. SVM classifier

The SVM is a machine learning algorithm which solves classification problems. A linear SVM is a classifier that searches for a hyper plane with the largest margin, which is why it is often known as a maximal margin classifier. Consider a binary classification problem consisting of N training examples. Each example is denoted by a tuple xi, i = 1,..., L. By convention, let y=+1 and y=-1 denote its class label. The decision boundary of a liner classifier can be written in the following form:

w.x+b=0 (3)

Where w and b are parameters of the model.
The training phase of SVM involves estimating the parameters w and b of the decision boundary from the training data. The parameters must be chosen in such a way that the following two conditions are met:

bi1:yi (w.xi+b)= 1 (4)
bi2: yi (w.xi+b)> 1 (5)
Since the objective function is quadratic and the constraints are linear in the parameters w and b, this is known as a convex optimization problem, which can be solved using the standard Lagrange multiplier method. Following is a brief sketch of the main ideas for solving the optimization problem.
Although the preceding conditions are also applicable to any linear classifiers, SVM imposes an additional requirement that the margin of its decision boundary must be maximal. Maximizing the margin, however, is equivalent to minimizing the following objective function:

F (w) =|w2|/2 (6)

The learning task in SVM can be formalized as the following constrained optimization problem:

Min|w2|/2 (7)

Since the objective function is quadratic and the constraints are linear in the parameters w and b, this is known as a convex optimization problem, which can be solved using the standard Lagrange multiplier method. Following is a brief sketch of the main ideas for solving the optimization problem.

$L_{P=}1/2|w|2 - \sum\lambda_i (y_i (w. x_i + b)-1)$ (8)

Where the parameters λ_i are called the Lagrange multipliers. The first term in the lagrangian is the same as the original objective function, while the second term captures the inequality constraints. To understand why the objective function must be modified, consider the original objective function given in Equation number (8). It is easy to show that the function is minimized when w = 0, a null vector whose components are all zeros. Such a solution, however, violates the constraints given in definition number because there is no feasible solution for b. The solutions for w and b are infeasible if they violate the inequality constraints; if $(y_i (w. x_i + b)-1) < 0$
The Lagrangian given in Equation number (8) incorporates this constraint by subtracting the term from its original objective function. Assuming that $\lambda_I > 0$, it is clear that any infeasible solution may only increase the value of the Lagrangian. To minimize the Lagrangian, we must take the derivative of Lp with respect to w and b an set them to zero.
Because the Lagrange multipliers are unknown, we still cannot solve for w and b. if definition number contains only equality instead of inequality constraints, then we can use the N equations from equality constraints along with equations number and number to find the feasible solution for w, b, and λ_i. Note that the Lagrange multipliers for equality constraints are free parameters that can take any values.
On way to handle the inequality constraints is to transform them into a set of equality constraints [9]. This is possible as long as the Langrange multipliers are restricted to be non-negatives. Such transformation leads to the following constraints on the Lagrange multipliers, which are known as the Karush-Kuhn-Tuker (KKT) conditions.

$$\lambda_i [y_i (w.x_i + b) - 1] = 0 \quad i = 1,...l \quad (9)$$

At first glance, it may seem that there are as many Lagrange multipliers as there are training instances. It turns out that many of the Lagrange multipliers become zero after applying the constraint given in equation number (9). The constraint states that the Lagrange multiplier λ_i must be zero unless the training instances x_i satisfies the equation $(w. x_i + b)-1= 1$. Such training instance, with $\lambda_i > 0$, lies along the hyperplanes bi1 or bi2 and is known as a support vector [8, 12, 13].
Training instances that do not reside along these hyperplanes have $\lambda_i > 0$.

© 2010, IJARCS All Rights Reserved

Solving the preceding optimization problem is still quite a daunting task because it involves a large number of parameters: w, b, and λ_i.

Support vector machines (SVMs) are particular classifiers which are based on the margin-maximization principle. They perform structural risk minimization, which was introduced to machine learning by Vapnik, and have produced excellent generalization performance. For nonlinear problems, SVMs use the kernel trick to produce nonlinear boundaries. The idea behind kernels is to map training data nonlinearly into a higher-dimensional feature space via a mapping function and to construct a separating hyper plane which maximizes the margin. The construction of the linear decision surface in this feature space only requires the evaluation of dot products
$K(\mathbf{x}_i, x_j) = \varphi(\mathbf{x}_i)\,\varphi(\mathbf{x}_j)$, where $K(\mathbf{x}_i, x_j)$ is called the kernel function [10, 11]. Three types of kernels polynomial kernel, RBF kernel and Linear Kernel are frequently used.

II. Parallel classifier

In this section we propose a parallel structure that used "one-vs- Other "method [7]. This method used for multi-class problems. If we have K class in the problem, then we partition the K classes into a two-class problem: one class contains True Output and another class contains False Output. Then we have five filters in this structure: one of them for Normal records and four filters for four groups of Attacks.

This structure has five filters that each filter uses a SVM, K-NN classifier. Each SVM or K-NN filters can detect one of attacks or normal records. Figure 1 shows the parallel structure.

For each SVM and K-NN classifier we calculate:

TP (True negative): The percentage of valid records that are correctly classified.

TN (True positive): The percentage of attack records that are correctly classified.

FP (False positive): The percentage of records that were incorrectly classified as attacks whereas in fact they are valid activities.

FN (False negative): The percentage of records that were incorrectly classified as valid activities whereas in fact they are attacks.

Then calculate:

Accuracy= (TP+TN)/ (TP+TN+FP+FN) (3)

FAR= FP/ (FP+TN) (4)

And we calculate average of Accuracy.

If one record is normal record, then first filter can make true or false output. If first filter make true then it's a normal record, else, four filters make own output. Each filter makes true or false output. If one of four outputs is true then, type of attack is defined. If two or more SVM, K-NN filters make True output, then system must be decision between them. For this section a decision block can decision between True outputs.

Decision block use Distance algorithm for Decision between true outputs, then appears the final result.

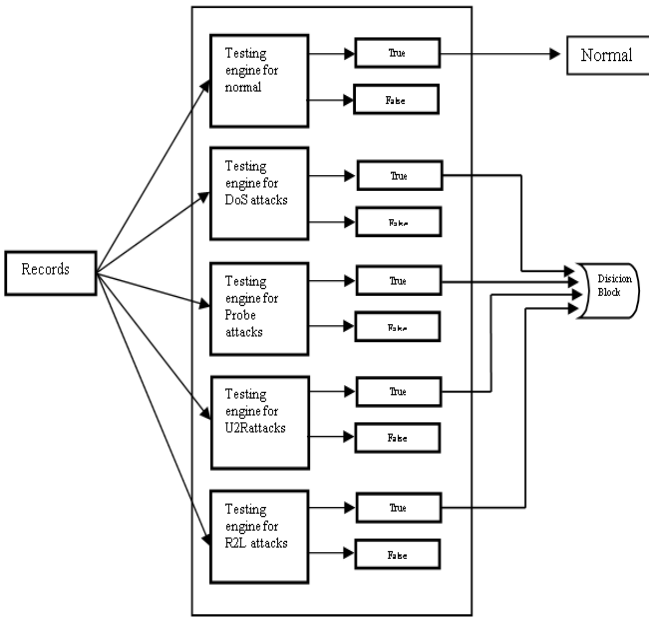


Figure 1: Parallel structure

III. EXPERIMENTAL RESULT

Using the 10% KDD data set we have 494,020 records that can be used to train the training engine. These training records consist of normal (non attack) records and known attack records distributed among the four attacks types: DoS, Probe, U2R and R2L.

We execute this structure in two methods: SVM and K-NN and Distance Algorithm for two methods are decision block.

Experiment Result for K-NN in parallel is shown in table2. We experiment k=1, 5, 10 for K-NN and the best result achieve when k=5. In this table we present Accuracy for normal and four groups of attacks, then calculate the average of Accuracy

Table II: Result of K-NN

Normal	DoS	R2L	U2R	Probe	Average of Accuracy
%73.95	%52.04	%69.35	%78.92	%90.52	%60.75

We use SVM for parallel structure with linear kernel, that make the best result then we calculate Accuracy for four groups of attacks and normal. The result present in table 3.

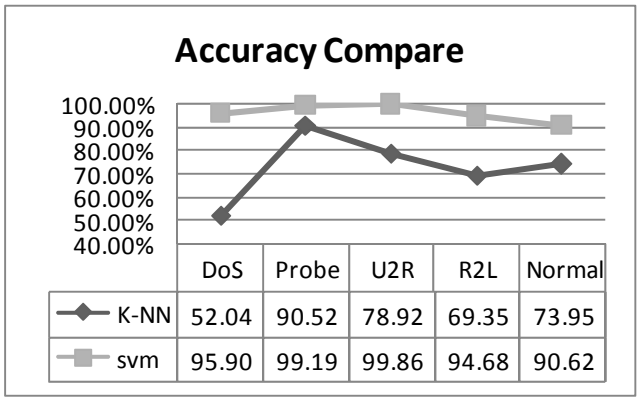
In this method we used genetic algorithm for feature selection and our result was improved.

Table III: Result SVM

Normal	DoS	R2L	U2R	Probe	Average of Accuracy
%90.62	%95.90	%94.68	%99.86	%99.19	%94.97

© 2010, IJARCS All Rights Reserved

We compare two results of table2 and table3 in figure 2.
This figure shows the SVM results are better than K-NN results.



	DoS	Probe	U2R	R2L	Normal
K-NN	52.04	90.52	78.92	69.35	73.95
svm	95.90	99.19	99.86	94.68	90.62

Figure 2: Comparing two group result

We also calculate FAR for each group. In our system we achieve FAR 0.13% for R2L attacks with Linear SVM and achieved overall accuracy 94.97% with Linear SVM.

CONCLUSION

Since the goal of this research was to improve the accuracy with using parallel structure, we have succeeded in achieving our target by using the parallel as an engine to classify the data accordingly. We achieve results superior to Chou in his PhD dissertation [1], where he achieved an accuracy of 87.33% for the R2L and we achieved 94.68% for R2L attacks with SVM classification method and he achieved an accuracy of 95.45% for the U2R and we achieved 99.86% for U2R attacks with SVM classification method. Hesham Altwaijry achieved an accuracy of 76.69% for the R2L and we achieved 94.68% for R2L attacks with SVM classification method. Our method is better than two methods because its accuracy is higher than them. Whit this approach we achieve accuracy of 94.97% for system. It's an average of five group of accuracy. Chou can achieved an FAR of 3.15% for the R2L and we achieved 0.13% for R2L attacks with SVM classification method and he achieved an accuracy of 3.13% for the U2R and we achieved 0.06% for U2R attacks with SVM classification method. Hesham Altwaijry achieved an overall accuracy of 82.1% and we achieved overall accuracy 94.97%.

REFERENCES

[1] Chou, Te-Shun, "Ensemble Fuzzy Belief Intrusion Detection Design" (2007). FIU Electronic Theses and Dissertations. Paper 6.
[2] Xiang C, Yong P C, Meng L S.2008. Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees, Pattern Recognition Letters 29 (2008) 918–924
[3] Hesham Altwaijry, Saeed Algarny, Bayesian based intrusion detection system, Journal of King Saud3-University - Computer and Information Sciences, Volume 24, Issue 1, January 2012, Pages 1-6
[4] KDD Cup 1999 Data[Online]. available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99Html.
[5] Hesham Altwaijry, Saeed Algarny, Multi-Layer Bayesian Based Intrusion Detection System, Proceedings of the World Congress on Engineering and Computer Science 2011 Vol II WCECS 2011, October 19-21, 2011, San Francisco, USA
[6] Levin, "KDD-99 Classifier Learning Contest LLSoft's Results Overview," ACM SIGKDD Explorations Newsletter, Volume 1, issue 2, pp. 67-75, January 2000.M
[7] Ding C, Dubchak I, multi-class protein recognition using support vector machines and neural network, 2000, Oxford Univ Press, vol.17 no.4 2001 pages 349-358
[8] Romero C, Garcia-Valdez M, Alanis A, A Comparative Study of Blog Comments SpamFiltering with Machine Learning Techniques, 2010, Soft Comp. for Recogn. Based on Biometrics, SCI 312, pp. 57–72.
[9] Law K, Kwok L, IDS False Alarm Filtering Using KNN Classifier, 2005, Information ecurity Applications Lecture Notes in Computer Science Volume 3325, 2005, pp 114-121
[10] B.E. Boser, I. Guyon, V. Vapnik," A training algorithm for optimal margin classifiers", computes. Learn. Theory pp.144–152, 1992.
[11] N. Cristianini, J. Shawe-Taylor, "An Introduction to Support Vector Machines", Cambridge University Press, Cambridge, 2000.
[12] B. Scholkopf, A.J. Smola," Learning with Kernels", MIT Press, Cambridge, MA, 2002.
[13] J. Shawe-Taylor, N. Cristianini, "Kernel Methods for Pattern Analysis", Cambridge University Press, Cambridge, 2004.

© 2010, IJARCS All Rights Reserved