Computer Communications 31 (2008) 4018–4025

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A lightweight web server anomaly detection method based on transductive scheme and genetic algorithms

Yang Li [a,b,*], Li Guo [b], Zhi-Hong Tian [b], Tian-Bo Lu [c]

[a] China Mobile Research Institute, Gate 2 Dacheng Plaza, No. 28 Xuanwumen West Street, Xuanwu District, Beijing 100053, China
[b] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[c] National Computer Network Emergency Response Technical Team/Coordination Center of China 100029, China

ARTICLE INFO

Article history:
Received 13 June 2007
Received in revised form 11 August 2008
Accepted 13 August 2008
Available online 22 August 2008

Keywords:
Network security
Web server anomaly detection
TCM-KNN algorithm
Genetic algorithm

ABSTRACT

World Wide Web (WWW) is one of the most popular applications currently running on the Internet and web server is a crucial component for this application. However, network anomalies especially Distributed Denial-of-Service (DDoS) attacks bombard web server, degrade its Quality of Service (QoS) and even deny the legitimate users' requests. Traditional network anomaly detection methods often lead to high false positives and expensive computational cost, thus unqualified for real-time web server anomaly detection. To solve these problems, in this paper we first propose an efficient network anomaly detection method based on Transductive Confidence Machines for *K*-Nearest Neighbors (TCM-KNN) algorithm. Secondly, we integrate a lot of objective and efficient anomalies impact metrics from the perceptions of the end users into TCM-KNN algorithm to build a robust web sever anomaly detection mechanism. Finally, Genetic Algorithm (GA) based instance selection method is introduced to boost the real-time detection performance of our method. We evaluate our method on a series of experiments both on well-known KDD Cup 1999 dataset and concrete dataset collected from real network traffic. The results demonstrate our methods are actually effective and lightweight for real-time web server anomaly detection.

1. Introduction

Web server is a critical and necessary component for Internet applications and web applications dominate the most part of network traffic nowadays, while they are suffering from a great deal of attacks especially Distributed Denial-of-Service (DDoS) attacks. DDoS is a large-scale, coordinated attack on the availability of services at a victim system or network resource. Therefore, DDoS significantly degrades service quality experienced by legitimate users, by introducing large delays, excessive losses, and service interruptions. The key point for DDoS defenses is to detect it as soon as possible and neutralize this effect, thereby quickly and fully restore quality of various services to levels acceptable by the users. Moreover, current evaluation methodologies measure DDoS damage superficially and partially by measuring a single traffic parameter, such as duration, loss or throughput, and showing divergence during the attack from the baseline case. These measures do not consider quality-of-service requirements of different applications and how they map into specific thresholds for various traffic parameters. They thus fail to measure the service quality experienced by the end users.

In essence, detecting attacks for web server can be classified as an intrusion detection problem. Intrusion detection system plays critical role of detecting various kinds of attacks. The main purpose of IDS is to find out intrusions among normal audit data and this can be considered as classification problem. The two basic methods of detection are signature based and anomaly based [1]. The signature-based method, also known as misuse detection, looks for a specific signature to match, signaling an intrusion. They can detect many or all known attack patterns, but they are of little use for as yet unknown attack methods. Most popular intrusion detection systems fall into this category. Another approach to intrusion detection is called anomaly detection. Anomaly detection applied to intrusion detection and computer security has been an active area of research since it was originally proposed by Denning. Anomaly detection algorithms have the advantage that they can detect new types of intrusions as deviations from normal usage. In this problem, given a set of normal data to train from, and given a new piece of test data, the goal of the intrusion detection algorithm is to determine whether the test data belong to "normal" or to an anomalous behavior. However, anomaly detection schemes suffer from a high rate of false alarms. This occurs primarily because previously unseen (yet legitimate) system behaviors are also recognized as anomalies, and hence flagged as potential intrusions. All in all, current anomaly detection methods developed very slowly for the following three reasons:

* Corresponding author. Tel.: +86 10138118203 45.
E-mail address: samsunglinux@163.com (Y. Li).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.08.009

(a) High false positives is difficult to avoid, which is mainly attributed to the complex network environment and poor adaptivities of current anomaly detection methods;
(b) Effective data mining and machine learning based anomaly detection methods are always followed by expensive computational cost;
(c) Anomaly detection methods especially data mining based methods greatly depend on the quality of training dataset for modeling, while in real network environment (even in the classical open data set), the quality of dataset is not so good as we expect. Therefore, how to choose the most qualified data for training is very important. However, current researches scarcely take into account this problem.

In this paper, we first propose a network anomaly detection scheme based on Transductive Confidence Machines for*K*-Nearest Neighbors (TCM-KNN) algorithm that has higher true positive rate, lower false positive rate than the traditional anomaly detection methods. Secondly, based on the efficient method, we mainly address its application in web server anomaly detection. The main contributions of this paper lie in the following aspects:

a) Firstly, we propose an efficient anomaly detection method for web server, it is much more effective than single, multiple traffic threshold based anomaly detection methods and traditional ones such as SVM, KNN, etc., since it introduces "strangeness" and "*p*-values" measures;
b) Secondly, we uses objective network anomalies measures in terms of the end users experiences of the Quality of Service (QoS) of web server, which actually captures the impact of attacks, flash crowds or any other fault network configurations on web server services. That ensures our TCM-KNN could detect web server anomalies more accurately than using the subjective traditional measures that are always assigned by the researchers empirically and often neglect the most important factor, that is, the end users experiences and evaluations towards the QoS of web server to determine the anomalies;
c) Finally, to boost the real-time anomaly detection performance of TCM-KNN for web server, we introduce Genetic Algorithm (GA) based instance selection mechanism to reduce the training dataset for TCM-KNN, thereby greatly reduce the computational cost and make it more suitable for near real-time anomaly detection in real network environment.

The rest of this paper is organized as follows. We outline the related work in Section 2. Section 3 discusses TCM-KNN algorithm and its application in web server anomaly detection in detail. We optimize TCM-KNN method using GA-based instance selection mechanism towards building more effective and lightweight detection method for web server in Section 4. Then, we report the relevant experiments and discussions in Section 5. Finally, we conclude our work in Section 6.

2. Related work

The related work about this paper can be briefly illustrated in two aspects: anomaly detection methods and DDoS impact metrics as well as its detection methods.

For the first aspect, in past several years, a lot of anomaly detection approaches attempt to build some kind of a model over the normal data and then check to see how well new data fits into that model. One approach uses a prediction model obtained by training decision trees over normal data [12], while others use neural networks to obtain the model [13] or Markov models [14] to detect novel attacks. However, the above so-called supervised anomaly detection approaches require a set of purely normal data from which they train their models. In the most circumstances, the data contains some intrusions buried within the training data, the algorithm may not detect future instances of these attacks because it will identify that they are normal. In the unsupervised anomaly detection problem, we are given a set of data where it is unknown which are the normal elements and which are the anomalous elements. The goal is to recover the anomalous elements. Unsupervised anomaly detection is a variant of the classical outlier detection problem. In practice, the cluster-based estimation algorithm, *K*-Nearest Neighbors algorithm and One-class SVM are proposed by authors in [15] for the unsupervised anomaly detection and it was demonstrated that they are superior to those traditional supervised anomaly detection approaches, especially the performance of One-class SVM algorithm both in detection True Positives (TP) and False Positives (FP). However, their TP and FP is not good as we expect (98% for TP and 10% for FP [15]).

As the latest work in the second aspect, many network detection schemes are based on packet based or flow based measures for performance reasons. This often leads to coarse grain attack detection signatures based on network level parameters such as the destination address and port [16,17]. Such coarse grain signatures are not always sufficient to block attack traffic appropriately. Moreover, most of the methods introduced so far are based on appearance of DDoS attack, such as spoofed source IP address, bandwidth distribution, attack packet pattern [20,21], etc. However, attacker can easily hide the appearance of attack traffic via packet resharping, thus escape the detection mechanisms. In contrast, Jelena, etc. recently proposed a novel measurement for DoS and DDoS towards the web applications from the perspective of end users in [7]. In essence, the new measurement could evaluate the impact of DDoS more accurately since they measure DDoS impact as a percentage of transactions that have not met their QoS requirements and aggregate this measure into several metrics that expose the level of service denial. However, based on their definition, they did not give any effective method or algorithm to substantially utilize it for DDoS detection in [7], therefore, to adopt it for real applications, it should be further improved and the first important problem to be seriously addressed, in our opinion, is how to integrate all the measurements into a reasonable DDoS detection algorithm such as TCM-KNN introduced in this paper.

3. Web server anomaly detection mechanism

3.1. Introduction to TCM-KNN principles

Transduction has been previously used to offer confidence measures for the decision of labeling a point as belonging to a set of pre-defined classes [2]. Transductive Confidence Machines (TCM) introduced the computation of the confidence using algorithmic randomness theory, a detail description of the theory can be found in [3]. It was proved that there exists a universal method of finding regularities in data sequences. Unfortunately, universal tests are not computable, and have to be approximated using non-universal tests called *p*-values [2]. In the literature of significance testing, the *p*-value is defined as the probability of observing a point in the sample space that can be considered more extreme than a sample of data. This *p*-value serves as a measure of how well the data fits the current classes. The smaller the *p*-value, the greater the evidence the point is an outlier. Users of transduction as a test of confidence have approximated a universal test for randomness (which is in its general form, non-computable) by using a *p*-value function called strangeness measure [3].

In the process of combining *K*-Nearest Neighbors (KNN) algorithm with TCM for network anomaly detection, we denote the sorted sequence (in ascending order) of the distances of point i from the other points with the same classification (normal class) as D_i. Also, D_{ij} will stand for the jth shortest distance in this sequence. We assign to every point a measure called the individual strangeness measure. This measure defines the strangeness of the point in relation to the rest of the points. In our case the strangeness measure for a point i belonging to a normal class is defined as

$$\alpha_i = \sum_{j=1}^{k} D_{ij} \tag{1}$$

where k is the number of neighbors used. The α computed for an anomaly (a point that does not belong to the normal class) will be the ratio between two large numbers (the distances from the point in question to those in the normal class are large). In some cases, this ratio will be small enough to be comparable to the α values for points already in the class, leading to false negatives [4–6,25].

Provided with the definition of strangeness, we will use Eq. (1) to compute the p-value as follows:

$$p(\alpha_{\text{new}}) = \frac{\#\{i : \alpha_i \geqslant \alpha_{\text{new}}\}}{n+1} \tag{2}$$

In Eq. (2), # denotes the cardinality of the set, which is computed as the number of elements in finite set. α_{new} is the strangeness value for the test point (assuming there is only one test point, or that the test points are processed one at a time), is a valid randomness test in the iid (independent and identical distribution) case. The proof takes advantage of the fact that since our distribution is iid, all permutations of a sequence have the same probability of occurring. If we have a sequence $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and a new element α_{new} is introduced then α_{new} can take any place in the new (sorted) sequence with the same probability, as all permutations of the new sequence are equiprobable. Thus, the probability that α_{new} is among the j largest occurs with probability of at most $\frac{j}{n+1}$.

From the above discussions, we can see that the p-value for the sequence $\{\alpha_1, \alpha_2, \ldots, \alpha_m, \alpha_{\text{new}}\}$, where $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ are the strangeness measures for the training points and α_{new} is the strangeness measure of a new test point with a possible classification assigned to it, is the value $p(\alpha_{\text{new}})$ [25].

3.2. Feature space for web server anomaly detection

Generally speaking, network anomalies concern on degrading the network services especially the web servers that service a great deal of legitimate users. As for web servers, the most serious impact on them is the effect of DDoS attacks since they usually make the necessary service quality for legitimate users unavailable, thereby reach their illegal personal or economic goals. Therefore, how to effectively detect anomalies especially DDoS attacks as quickly as possible and make corresponding countermeasures to alleviate the negative impact of attacks on QoS of web servers is a critical problem should be seriously considered.

It is no strange the effectiveness of anomaly detection is based on both algorithms and selected features that could utmost differentiate the normal from anomalous traffic. First of all, we should select the most effective features. As stated by authors in [7], traditional anomaly detection methods especially those for web server anomaly detection usually take use of many subjective parameters (e.g., the packet length, the packets counts in a certain time slot) proposed by researchers as input features to detect anomalies, they all unfortunately ignored the objective aspects, that is, the end users experiences toward the current network traffic, since it is a user that indeed perceives some anomalies such as large request-response delay in interactive traffic, low-quality audio and image due to packet loss in media and gaming traffic, and large duration of non-interactive transactions such as email transfer because of the impact of anomalies such as DDoS attacks on network traffic. Therefore, in this paper, we would adopt the effective objective measures proposed in [7] as our features for TCM-KNN algorithm to detect anomalies in network since their efficacies have been proved by the authors. Moreover, it is worth mentioning that because our TCM-KNN algorithm is not a single threshold based method, it combines a lot of parameters, thus it is more efficient than what is illustrated in [7] since the authors only simply use some thresholds to detect anomalies.

In more detail, for each web transaction (it represents higher-level tasks whose completion is meaningful to a user, such as: browsing one Web page, downloading a file or having a VoIP conversation) as mentioned in [7], we also measure five parameters: (1) one-way delay, (2) request/response delay, (3) packet loss, (4) overall transaction duration and (5) delay variation (jitter). Jointly, these parameters capture a variety of application QoS requirements [8,9]. The first thing for us to detect DDoS attacks according to these measures is mapping them into a feature vector. Using these feature vectors, we can build a pattern model for normal perceptions of the QoS of web server by end users. Therefore, we would acquire a lot of points (namely feature vectors) as a training dataset for TCM-KNN.

3.3. TCM-KNN algorithm for web server anomaly detection

From the discussions stated in Section 3.1, in general classification cases, using the α values, we can compute a p-value for the new point for the normal classes. This gives us a way of testing the fitness of point s for the only class with a confidence of at least $\delta = 1 - \tau$, and it thus provides us with the web server anomaly detection method. Selecting a confidence level δ (usually 95%), we can test if p *les* τ, in which case, we can declare the point an anomaly. Otherwise, we declare it's normal. The detailed algorithm is depicted in Fig. 1. Because it does not require building a classifier, thus the parameters for TCM-KNN are only the k for the number of nearest neighbors, τ for the threshold of TCM-KNN. They were empirically selected as 300 and 0.05, respectively, in our experiments.

Based on the above descriptions, the web server anomaly detection tasks could be performed in two phases as illustrated in Fig. 2. In the training phase, we collect a lot of points formed by the above five parameters. These points represent the normal web server status (i.e., the QoS experienced by the end users), therefore, when encountering DDoS attack, the web server status will change and the relevant points would be different from those normal points. Moreover, with the employment of our TCM-KNN algorithm, in the detection phase, the abnormal points could be diagnosed successfully and we may claim the web server is under DoS attack and the corresponding countermeasures should be taken, thus the effect of attack on web server and its service would be effectively alleviated and it will greatly ensure the QoS of web server.

In essence, TCM-KNN is an excellent anomaly detection algorithm, which combines transductive learning method with classical KNN classification algorithm effectively, and thus does well in distinguishing normal and abnormal traffic with high confidence for network anomaly detection domain, compared to the traditional methods such as KNN score, One-class SVM, etc. Unlike traditional methods in data mining, it can offer measures of reliability to individual points one at a time without a previously built model, and directly uses the points (normal points) in hand to determine the new arrival is normal or not. The core concepts, p-value and strangeness serve as effective measures of how well the data belongs to a certain class, which make TCM-KNN more effective in network anomaly detection.

Parameters: k(the nearest neighbors to be used), n(size of training dataset), τ (preset threshold), s (instance to be determined)

for $i = 1$ to m {

calculate D_i for each one in training dataset and store;

calculate strangeness α according to Equation (1) for each one in training dataset and store;

}

calculate the strangeness for s according to Equation (1);

calculate the p-values for s according to Equation (2);

if ($p \leq \tau$)

determine s as anomaly with confidence $1-\tau$ and return;

else

claim s is normal with confidence $1-\tau$ and return;

Fig. 1. TCM-KNN algorithm for web server anomaly detection.

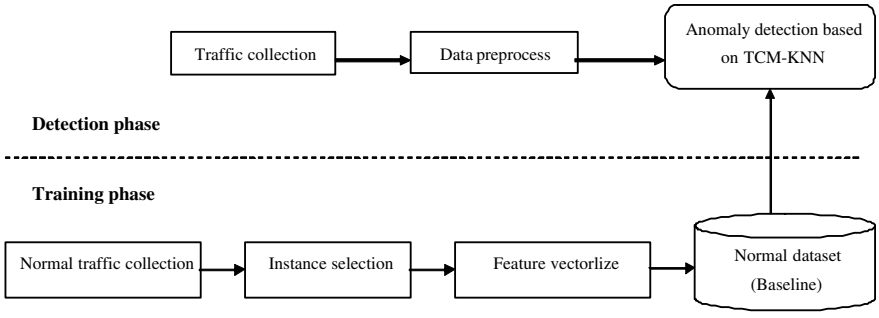

Fig. 2. Illustration of web server anomaly detection.

In addition, it is worth mentioning that in this paper we would not clearly differentiate the anomalous traffic caused by attacks and that caused by flash crowds or fault network configuration, because in the point view of end users, they may experience longer request/response delay and one-way delay, etc. when the web server encounters real network attack, temporary flash crowds, fault network configuration or any other anomalies. Moreover, the main goal of this paper is to discuss the relevant technique aiming to detect such anomalies as accurately and quickly as possible, the only differences only lie in the corresponding responses to some kinds of anomalies, but not in the detection phase. Therefore, to emphasis the reasonability and availability of our method presented in this paper, we always use DDoS attack in our descriptions and discussion in this paper.

We may take a look at the computational cost of the discussed anomaly detection algorithm. Since the method requires finding k nearest neighbors on each class, we need $O(n)$ distance computations, per each point to be diagnosed, where n is the number of data points in the normal data set. Hence, to diagnose s points, the complexity would be $O(ns)$. Moreover, to find out the k nearest neighbors for the normal data set, we require $O(n^2)$ comparisons. We observe that this step is done off-line and only once, before the detection of anomalies starts. However, if n is very large, the off-line computation may still be too costly. Therefore, the size of training dataset is a key point that influences the detection performance of our method.

In the next section, in terms of its good performances demonstrated in computer security domain, we will introduce the effective evolutionary algorithm – Genetic Algorithm (GA) to select the most important and contributed instances for the training set of anomaly detection, which would greatly boost the real-time detection performance of our proposed method and makes it more suitable for the real network environment.

4. Optimizations for web server anomaly detection

4.1. Introduction to Genetic Algorithm (GA)

Genetic Algorithms (GA) are optimization algorithms based on natural genetics and selection mechanisms [10]. The basic concept of GA is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. To apply genetic algorithms to a particular problem, it has to be decomposed in atomic units that correspond to genes. The genetic information (chromosome) is represented by a bit string and sets of bits encode the solution. The bit string may be of variable length. Then individuals can be built with correspondence to a finite string of genes, and a set of individuals is called a population. A criterion needs to be defined: a fitness function F which, for every individual among a population, gives $F(x)$, the value which is the quality of the individual regarding the problem we want to solve.

Genetic algorithm uses three operators: recombination, selection and mutation. The recombination process is called crossover in this algorithm, which it is a sexual operation that creates

two offspring strings from two parent strings copying selected bits from each parent. The selection operation is repeated as often as desired usually until the new generation is completed. Mutation is carried out by randomly changing the value of a single bit (with small probability) to the bit strings. In these algorithms individuals are chosen with a probability according to their objective function values. Some of these selected individuals, considered as the best individuals, are carried forward into the next generation population intact. This operation is known as elitism.

From the above descriptions about GA, we think GA is very suitable to be used to fulfill instance selection task for TCM-KNN algorithm. In fact, the instance selection task for TCM-KNN is an evolutionary process in nature, it does need to select the most contributed instances and omit the useless ones in virtue of three operators (named recombination, selection and mutation) and the fitness function (to be discussed later). Moreover, GA has gained great successful applications in several domains. For example, GA is used for training data selection in radial based function networks. In the domain of network intrusion detection, genetic algorithms have been used in a number of ways. Some approaches [22] have used genetic algorithms directly to derive classification rules, while others [23,24] use genetic algorithms to select appropriate features or determine optimal parameters of related functions, while different data mining techniques are then used to acquire the rules. In terms of the good performance of GA in so many areas discussed above, we will adopt it to fulfill the task of selecting high quality instances as training dataset for TCM-KNN in web server anomaly detection.

4.2. Instance selection based on GA

To apply GA to instance selection for the training dataset of TCM-KNN, two important issues should be addressed: the specification of the representation of the solutions and the definition of the fitness function.

(*a*) *Representation:* For TCM-KNN, training dataset is denoted as *TR* with instances. The search space associated with the instance selection of *TR* is constituted by all the subsets of *TR*. Then, the chromosomes should represent subsets of *TR*. This is accomplished by using a binary representation. A chromosome consists of genes (one for each instance in *TR*) with two possible states: 0 and 1. If the gene is 1, then its associated instance is included in the subset of *TR* represented by the chromosome. If it is 0, then this does not occur. After running GA algorithm, the selected chromosomes would be the reduced training dataset for TCM-KNN.

(*b*) *Fitness function:* Let be a subset of instances of *TR* to evaluate and be coded by a chromosome. For TCM-KNN based anomaly detection task, three measures should be seriously considered: detection rate, false positive rate and the percentage of training dataset reduction. Therefore, we define a fitness function that combines the three values: the detection rate (*detect_rate*) associated with, false positive rate (*fal_rate*), and the percentage of reduction (*reduce_rate*) of instances of with regards to *TR*. The TCM-KNN classifier is used for measuring the detection rate and false positive rate.

$$F(x) = C * (\det ect\_rate - fal\_rate) + (1 - C) * reduce\_rate \quad (3)$$

where, *reduce_rate*is defined as:

$$reduce\_rate = \frac{|TR| - |S|}{|TR|} * 100 \quad (4)$$

also, |TR| and |S| stand for the number of the original training dataset and of the reduced training dataset using GA. *C* is an adjustment constant and it always set by experiences. In our next experiments, we set is 0.6 empirically. The objective of the GA is to maximize the fitness function defined, i.e., maximize the detection rate and minimize the number of instances obtained as well as the false positive rate. We use TCM-KNN to evaluate the fitness of a chromosome.

Besides the fitness function and the representation problems to be serious considered when we apply it to anomaly detection, the three operators (namely recombination, selection and mutation) also should be considered in GA. Learning form the published literatures, a great deal of searching strategies for GA have been proposed and they include various of different implementation of the three operators. Four well-known models will be used in this paper. The first two are GGA (Generational Genetic Algorithm) and SGA (Steady-State Genetic Algorithm), the third is CHC (heterogenious recombination and cataclysmic mutation) adaptive search algorithm and the last is PBIL (Population-Based Incremental Learning). The principles about them can be found in [11]. In the experiments, we will adopt them for searching strategy and give the detailed relevant running results in Section 6.

5. Experimental results

To demonstrate the effectiveness of TCM-KNN based anomaly detection method, we first compare it with the most well-known unsupervised anomaly detection methods proposed in [8] both in true positives (TP) and false positives (FP) on the classic KDD Cup 1999 dataset. Secondly, we testify its efficiency and effectiveness in real network environment when applied to web server anomaly detection. Finally, we report the results when conducting GA based instance selection experiment to optimize TCM-KNN to enhance its computational efficiency for real-time anomaly detection.

5.1. Experimental results on KDD Cup 1999

As our experimental dataset, KDD 99 dataset contains 24 different types of attacks (including neptune, land, ipsweep, buffer overflow, etc.) that are broadly categorized in four groups such as Probes, DoS (Denial of Service), U2R (User to Root) and R2L (Remote to Local). The packet information in the original tcpdump files were summarized into connections. This process is completed using the Bro IDS, resulting in 41 features for each connection. Therefore, each instance of data consists of 41 features and each instance of them can be directly mapped into the point discussed in TCM-KNN algorithm. In the contrast experiments, we extracted the "normal" instances as training dataset (about 97,278) and the "abnormal" instances (about 4768 and all sampled from the dataset and included all the four attack types) as test dataset. We adopted 10-fold cross validation method to make the experiment. The experimental parameters were set as follows. For the unsupervised anomaly detection algorithms, we set their parameters as in [8] for the convenience of comparison. The detail is, for cluster-based algorithm, we set the width w of the fixed-width clustering to be 40. For the KNN algorithm, k was set to 10,000 and also the One-class SVM algorithm, we set $\nu = 0.01$, $\sigma^2 = 12$. For our method, we set k 300 and the confidence measure $\delta = 0.95$ (means $\tau = 0.05$) (the two most optimized settings were empirically selected from a series of experiments). Fig. 3 shows an ROC (Receiver Operating Characteristic) curve depicting the results. It is clear that our method demonstrates higher TP and lower FP than the other three methods.

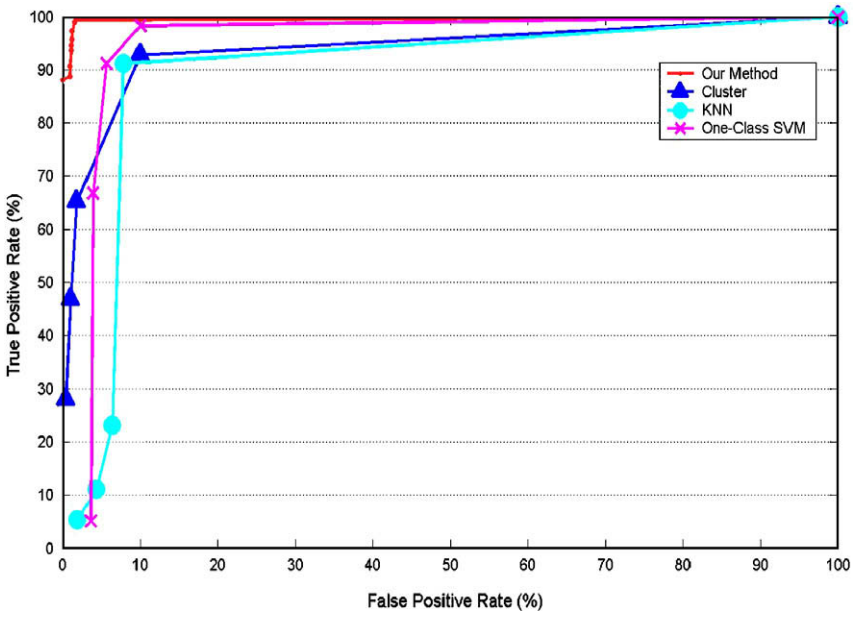

Fig. 3. ROC curves showing the performance of our method and other three algorithms.

5.2. Real network experimental environment

To test and evaluate the efficiency and effectives of our TCM-KNN method when applying it to web server anomaly detection, we also performed a series of experiments on a real research network. We selected a Gigabit Ethernet link between a large ISP and a college whose end users is about 10,000. The experimental environment contains both standard network services like web traffic, ftp traffic, peer-to-peer application traffic, online games as well as live audio and video streams, therefore, the more it is like the real network environment, the more persuasive results we may claim. That is to say, we want to demonstrate the effectiveness of our anomaly detection method is independent of the experimental environment.

We setup a web server located in the college running apache http service (version 2.2) on Linux platform (Red Hat Enterprise Edition 4, kernel 2.6.9-42.0.2.EL). We conducted many experiments over several days during busy hours and with background traffic generated from more than 5000 hosts of the college. In the experiments, the attackers can access the victim web server and they launched well-known DDoS attacks using a series of DDoS tools such as Stacheldraht [18] and TFN2K [19]. We performed many flooding attacks with spoofed IP's like SYN Floods, UDP and ICMP attacks. Moreover, we placed the detection engine in a monitor host to periodically collect the web server status info as illustrated in Section 4, as well as determine if the web server encounters anomalies by using our TCM-KNN method. The network topology of our experiments is shown in Fig. 4.

5.3. Data preprocessing

On one hand, in the controlled real network environment, the periodical collection process results in a training dataset for our TCM-KNN method, whose size is 51840 and they are all normal status before the web server encounters DDoS attacks. The dataset contains a series of five tuples (one-way delay, request/response delay, packet loss, overall transaction duration, delay variation). In other words, each point (feature vector) in the training dataset is 5-tuple. Meanwhile, after launching many DDoS attacks, we also collected a lot of five tuples (about 5600) that represent the anomalous statistical status of the web server experienced by the end users. We will use them to testify the detection function of our TCM-KNN anomaly detection engine.

Before beginning our experiments, we preprocessed the dataset. Firstly, we normalized the dataset. In order to avoid one value will dominate another for the numerical data, they were normalized by replacing each attribute value with its distance to the mean of all the values for that attribute in the instance space. In order to do this, the mean and standard deviation vectors must be calculated:

$$\text{mean}[j] = \frac{1}{n}\sum_{i=1}^{n}\text{instance}_i[j] \tag{5}$$

$$\text{standard}[j] = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\text{instance}_i[j] - \text{mean}[j])^2} \tag{6}$$

From this, the new instances can be calculated by dividing the difference of the instances with the mean vector by the standard deviation vector:

$$\text{newinstance}[j] = \frac{\text{instance}[j] - \text{mean}[j]}{\text{standard}[j]} \tag{7}$$

This results in rendering all numerical attributes comparable to each other in terms of their deviation from the norm, which is exactly what we are seeking in anomaly detection.

Moreover, the experiments employed Euclidean distance metric to evaluate the distance between two points. Although there are many other distance metrics available, such as Minkowsky, Mahalanobis, Camberra, Chebychev, we adopt the commonly used Euclidean distance metric only for convenience. We take it for granted that selecting anyone of them for experiments might have little effect on evaluating the experimental results and we will adopt the others in our future work for distance calculations.

The Euclidean distance metric is defined as Eq. (8), where Y_1 and Y_2 are two feature vectors, Y_{ij} denotes the jth component of Y_i and $|Y_i|$ denotes the length of vector Y_i:

$$\text{distance}(Y_1, Y_2) = \sqrt{\sum_{j=1}^{|Y_1|}(Y_{1j} - Y_{2j})^2} \tag{8}$$

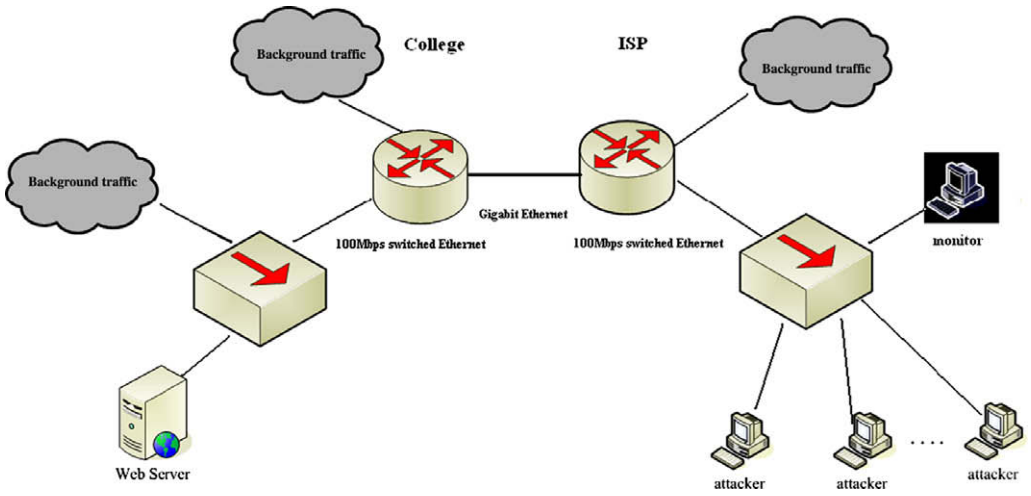

Fig. 4. Network topology for real experimental environment.

5.4. Experimental results in real network environment

To compare the results with that using the well-known unsupervised anomaly detection methods (One-class SVM, KNN and cluster) in web server anomaly detection, we also use them to detect and the comparison results is depicted in Table 1. We found their detection results are relatively worse, since the best results come from One-class SVM (detection rate 89.85% and 17.83% for false positive rate), and they demonstrated much worse performance for web server anomaly detection using concrete collected dataset from real network traffic than using KDD Cup 1999 dataset to make experiments [6], while our TCM-KNN algorithm retains good detection performance both using real network traffic and synthetic KDD Cup 1999 dataset. Table 2 describes the experiment configurations for various GAs aiming to boost the real-time detection performance of TCM-KNN. They include GGA, SCA, CHC and PBIL searching strategies. Table 3 shows the relevant experimental results. For each row (bold part denotes the most ideal results for each column) in this table, the results were acquired by selecting the average value of all kinds of measures (including detection rate (also named true positive rate, TP), false positive rate (FP) and reduced percentage) for each algorithm. We found TCM-KNN+CHC offered us the best results. It presents the best detection rate, false positive rate and the second best reduction rate among these algorithms. TCM-KNN+PBIL algorithm presents us the best reduction rate.

Moreover, to verify the effectiveness of instance selection for TCM-KNN, we compared the detection performance and computational cost (consuming time) of original TCM-KNN and the optimized TCM-KNN. All the results are evaluated in our monitor machine (see Fig. 4) with Pentium(R) 4 CPU processor 3.00 GHz, 1 GB memory and 80 GB hard disk, running in Linux platform.

In Table 4, building time denotes the time for calculating the strangeness and *p*-values of training dataset and detection time represents the average time for diagnosing a test data in on-line style. We can see that the average building time for each instance is reduced from 0.4286 s (22218.62/51840) to 0.1663 s (363.86/2188), and detection time from 0.4164 to 0.1397 s.

5.5. Discussions

From the results presented in Section 5.4, we found that we could determine the anomalous points with accuracy of 100% and only 3.18%false positives in the real network environment. We think these ideal results attribute to two reasons: one is the adoption of end users' experiences towards the QoS of web server to evaluate the effect of DDoS attack on QoS of web server. The other is the application of the effective TCM-KNN anomaly

Table 1
Comparison results between various detection algorithms in real network environment

Detection algorithm	TP (%)	FP (%)
TCM-KNN	**100**	**3.18**
One-class SVM	89.85	17.83
KNN	80.72	12.39
Cluster	62.57	15.37

Table 2
Experiment configurations for various GAs

Searching strategy	Parameters
GGA	*Pm* = 0.001, *Pc* = 0.6, *pop* = 100, *Eval* = 10000
SGA	*Pm* = 0.001, *Pc* = 0.1, *pop* = 100, *Eval* = 1000
CHC	*Pop* = 100, *Eval* = 10000
PBIL	*Pm* = 0.02, *Pc* = 0.1, *pop* = 100, *Eval* = 10000, *LR* = 0.1, *Mult_shift* = 0.05, *Negative_LR* = 0.075

Table 3
Experimental results for GA based instance selection mechanism

Algorithm	TP (%)	FP (%)	Reduction (%)
TCM-KNN+GGA	89.35	7.46	95.54
TCM-KNN+SGA	87.48	11.28	88.33
TCM-KNN+CHC	**99.38**	**3.87**	95.78
TCM-KNN+PBIL	98.73	5.47	**96.33**

Table 4
Performance optimization results after instance selection

	Building time (s)	Detection time (s)	TP (%)	FP (%)
TCM-KNN (original)	22218.62	0.4164	100	3.18
TCM-KNN (optimized)	**363.86**	**0.1397**	99.38	3.87

detection algorithm to detecting anomalies for web server. Moreover, we found TCM-KNN algorithm achieved 100% TP in our experiment, while the previous work proposed in [6] had achieved no better than 90%. We think it is not very surprising. On one hand, we have thoroughly analyzed and compared these work with TCM-KNN in [6], the experimental results clear showed the prevalence of TCM-KNN over the others; On the other hand, as discussed in [6], the first three algorithms detect anomalies with the help of simple distance calculation or space division, which are less suitable for the anomaly detection than TCM-KNN algorithm, because TCM-KNN effectively combines distance calculation and the contributions of all the training data points to the anomaly detection by using the two key concepts: strangeness and p-value.

In addition, we observed these improvements resulted from the application of GAs are reasonable. On one hand, we adopted GA based instance selection to sharply reduce the size of original training dataset from 51840 to 2188, the reduction rate is about 95.78% and it contributed to saving 98.36%, building time. On the other hand, attributed to the obvious reduction of training dataset, the detection time for each point is also saved about 66.45% because as illustrated in TCM-KNN anomaly detection algorithm (see Fig. 1), with the reduction of training dataset, the computational cost of calculating strangeness and p-values for each point to be diagnosed is also greatly reduced. Moreover, although the reduction rate is obvious, the TP still keeps high and the FP is still manageable. Hence, all the results substantially evident that our TCM-KNN algorithm could be effectively optimized as a lightweight anomaly detection method suitable for near real-time detection and is suitable for real large-scale network traffic environment.

It is worth mentioning here that although the training phase of our TCM-KNN method for web server anomaly detection is offline, the detection phase is online and its computational cost is much lower than that of traditional data mining or machine learning methods because of its simple mathematical calculations both in strangeness and p-values, thus it is particularly suitable for real-time anomaly detection. To accurately and effectively perform anomaly detection task, periodically updating the training dataset to catch the web server status is a necessity.

6. Conclusions and future work

This paper presents our work focusing on how to effective detect anomalies especially DDoS attack against web server based on TCM-KNN algorithm and hence ensure the QoS of web server. The experimental results demonstrate it is a promising job and needs further improvements.

For our future work, the first work is we are attempting to detect anomalies for web server combining active methods with passive detection methods. The former is based on users' perceptions toward the web server status as discussed in this paper. The latter concerns on sniffing the incoming/outgoing network traffic of web server and make a lot of statistics. Then, a data fusion mechanism would be used according to the results of the above two detection methods to determine whether the web server is under attack. We think the comprehensive and data fusion based detections are more robust and effective than single active or passive detection especially in reducing the unnecessary false positives. By doing these, we hope more accurately and reasonably detect the anomalies for web server, hence ensure the QoS of web server. Moreover, we will further emphasis our research topic in detection and response to the various anomalies for web servers that we have not addressed yet in our current work, since every anomaly has its different impact on the QoS of web server, we should differentiate them and make corresponding countermeasures.

References

[1] D.E. Denning, An intrusion detection model, IEEE Transactions on Software Engineering SE-13 (1987) 222–232.
[2] A. Gammerman, V. Vovk, Prediction algorithms and confidence measure based on algorithmic randomness theory, Theoretical Computer Science (2002) 209–217.
[3] M. Li, P. Vitanyi, Introduction to Kolmogorov Complexity and Its Applications, second ed., Springer-Verlag, 1997.
[4] K. Proedru, I. Nouretdinov, V. Vovk, A. Gammerman, Transductive confidence machine for pattern recognition, in: Proc. 13th European Conference on Machine Learning, 2002, pp. 381-390.
[5] B. Daniel, D. Carlotta, P.R. James, Detecting outliers using transduction and statistical testing, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, 2006, pp. 55–64.
[6] Y. Li, B.X. Fang, L. Guo, Y. Chen, Network anomaly detection based on TCM-KNN algorithm, in: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS' 2007), ACM, 2007, pp. 13–19.
[7] J. Mirkovic, P. Reiher, S. Fahmy, R. Thomas, A. Hussian, S. Schwab, C. Ko, Measuring Denial of Service, QoP'06, ACM, 2006, pp. 53–58.
[8] J. Ash, M. Dolly, C. Dvorak, A. Morton, P. Taraporte, Y.E. Mghazli, Y.1541-QOSM – Y.1541 QoS Model for Networks Using Y.1541 QoS Classes, NSIS Working Group, Internet Draft, May 2006 (Work in progress).
[9] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, M. Claypool, The effects of loss and latency on user performance in unreal tournament 2003, in: Proceedings of ACM Network and System Support for Games Workshop (NetGames), September 2004.
[10] Laetitia Jourdan, Clarisse Dhaenens, El-Ghazali Talbi, A Genetic algorithm for feature selection in data-mining for genetics, in: Proceedings of the 4th Metaheuristics International Conference, 2001, pp. 29–33.
[11] Jose Ramon Cano, Francisco Herrera, Manuel Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Transactions on Evolutionary Computation, 2006, pp. 561–575.
[12] W. Lee, S.J. Stolfo, Data mining approaches for intrusion detection, in: Proceedings of the 1998 USENIX Security Symposium, 1998.
[13] A. Ghosh, A. Schwartzbard, A study in using neural networks for anomaly and misuse detection, in: Proceedings of the 8th USENIX Security Symposium, 1999.
[14] M. Mahoney, P. Chan, Learning nonstationary models of normal network traffic for detecting novel attacks, in: Proceeding of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002, pp. 376–385.
[15] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data, Applications of Data Mining in Computer Security, Kluwer, 2002.
[16] Haining Wang et al., Detecting SYN flooding attacks, in: IEEE Infocom'2002, pp. 1530–1539.
[17] T. Peng et al., Protection from distributed denial of service attack using history-based ip filtering, in: IEEE ICC 2003.
[18] D. Dittrich, The stacheldraht distributed denial of service attack tool. Available from: <http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>.
[19] J. Barlow, W. Thrower, TFN2K – an analysis. Available from: <http://packetstormsecurity.com/distributed/TFN2k_Analysis-1.3.txt>.
[20] Haining Wang, Cheng Jin, Kang G. Shin, Defense against spoofed IP traffic using hop-counter filtering, IEEE/ACM Transactions on Networking 15 (1) (2007) 40–53.
[21] D. Yau, J. Lui, F. Liang, Y. Yang, Defending against distributed denial-of-service attacks with max–min fair server-centric router throttles, IEEE/ACM Transactions on Networking 13 (1) (2005) 29–42.
[22] W. Li, Using genetic algorithm for network intrusion detection, C.S.G. Department of Energy, 2002, pp. 1–8.
[23] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[24] F. Pernkopf, P.O. Leary, A Genetic algorithm for feature selection in data-mining for genetics, in: Proc. of 4th Metaheuristics International Conference, 2001, pp. 29–33.
[25] Yang Li, Li Guo, An active learning based TCM-KNN algorithm for supervised network intrusion detection, Computers & Security 26 (7–8) (2007) 459–467.