Expert Systems with Applications 38 (2011) 3492–3498

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers

Ming-Yang Su *

Department of Computer Science and Information Engineering, Ming Chuan University, 5 Teh Ming Road, Gwei Shan District, Taoyuan 333, Taiwan

ARTICLE INFO

Keywords:
KNN (k-nearest-neighbor) classification
Genetic algorithm
NIDS (network intrusion detection system)
Network security
DoS attacks
Feature selection
Feature weighting

ABSTRACT

This study proposed a method which can detect large-scale attacks, such as DoS attacks, in real-time by weighted KNN classifiers. The key factor for designing an anomaly-based NIDS is to select significant features for making decisions. Not only is excellent detection performance required, but real-time processing is also demanded for most NIDSs. A good feature selection policy, which can choose significant and as few as possible features, plays a key role for any successful NIDS. The study proposed a genetic algorithm combined with KNN (k-nearest-neighbor) for feature selection and weighting. All initial 35 features in the training phase were weighted, and the top ones were selected to implement NIDSs for testing. Many DoS attacks were applied to evaluate the systems. For known attacks, an overall accuracy rate as high as 97.42% was obtained, while only the top 19 features were considered. For unknown attacks, an overall accuracy rate of 78% was obtained using the top 28 features.

1. Introduction

E-commerce systems are based upon Internet use, which provides open and easy communications on a global basis. Since the Internet is unregulated, unmanaged and uncontrolled, it introduces a wide range of risks and threats to the systems operating on it. This is the reason that the network intrusion detection systems (NIDSs) have been emerging recently. NIDSs are traditionally divided into two broad categories: misuse detection and anomaly detection. Misuse detection aims to detect known attacks by characterizing the rules that govern these attacks. Thus, rules update is particularly important, and consequently, new definitions are frequently released by NIDS vendors. However the rapid emergence of new vulnerabilities makes misuse detection difficult to trust. Anomaly detection is designed to capture any deviation from the profiles of normal behavior patterns. Anomaly detection is much more suitable than misuse detection to detect unknown or novel attacks, but it has the potential to generate too many false alarms. Therefore, this study proposed a system for anomaly detection.

Most NIDSs emphasize on effectiveness but neglect efficiency, especially for anomaly-based NIDSs. Usually, effectiveness is measured by detection rate, false alarm rate, etc, and efficiency is measured by response time while an attack occurs. Since too many features for an anomaly-based NIDS would not necessarily guarantee good performance, it certainly delays the detection engine's ability to make a decision. Thus, how to select fewer but significant features becomes vital. Furthermore, features should be weighted because their contributions to correct classification are different from each other. That is the goal of the paper. Since DoS/DDoS (Denial-of-Service/Distributed DoS) attacks are prevalent and becoming one of the main threads to E-commerce systems, this system was evaluated by DoS/DDoS attacks.

For an anomaly-based NIDS, the most difficult part is to present the normal profile, which depends on the policy of feature weighting and selection. In past studies, many good anomaly-based NIDSs have focused on system architectures or detection engine designs (e.g. Carl, Kesidis, Brooks, & Rai, 2006; Gavrilis & Dermatas, 2005; Kulkarni & Bush, 2006; Wang, Zhang, & Shin, 2004); only few of them have focused on the feature weighting and selection, such as Mukkamala and Sung (2002), Sung and Mukkamala (2003), Lee, Chung, and Shin (2006), Abbes, Bouhoula, and Rusinowitch (2004), Stein, Chen, Wu, and Hua (2005), Hofman, Horeis, and Sick (2004), Middlemiss and Dick (2003), Liao and Rao Vemuri (2002). Most of them have applied KDD CUP99 dataset for experiments. In order to promote the comparison of different works in IDS (intrusion detection system) area, the Lincoln Laboratory at MIT, under the Defense Advanced Research Project Agency (DARPA) and Air Force Research Laboratory sponsorship, constructed and distributed the first standard dataset for evaluation of computer network IDS (DARPA, XXXX) in 1998. In 1999, the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining with the purpose of demonstrating the learning contest,

* Tel.: +886 3 3507001; fax: +886 3 3593874.
E-mail address: minysu@mail.mcu.edu.tw

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.08.137

collected and generated TCP dump data provided by the aforementioned DARPA (XXXX) in the form of train-and-test sets. The above dataset is named as KDD CUP99 dataset (KDD CUP, 1999a).

Mukkamala and Sung (2002) applied the technique of SVMs (Support Vector Machines) to rank the 41 features provided by KDD CUP99 dataset (KDD CUP, 1999b). Mukkamala and Sung ranked the features again by both SVMs and neural networks in Sung and Mukkamala (2003). Lee et al. (2006) discussed the feature selection based on genetic algorithm combined with relief tree and genetic algorithm combined with Naïve Bayesian network. They also used KDD CUP99 dataset for experiments. Abbes et al. (2004) and Stein et al. (2005) applied decision trees to design their detection engines. Tree nodes were selected by genetic algorithm in Abbes et al. (2004), and information gain mixed with gain ratio and Gini index in Stein et al. (2005). A self-created dataset was experimented with in Abbes et al. (2004), while KDD CUP99 dataset was used in Stein et al. (2005). Hofman et al. (2004) applied genetic algorithm combined with an RBF (radial basis function) network to feature selection, and took seven attacks out of KDD CUP99 dataset for an experiment. Finally, Middlemiss and Dick (2003) and Liao and Rao Vemuri (2002) both proposed a genetic algorithm combined with KNN for feature selection. The KDD CUP99 TCPDUMP was experimented with in Middlemiss and Dick (2003), while 1988 DARPA BSM audit data (DARPA, XXXX) was experimented with in Liao and Rao Vemuri (2002); here, BSM represents audit logs generated on the Sun machine using Solaris Basic Security Module (BSM). However, in Middlemiss and Dick (2003) the details about genetic algorithm and KNN were not mentioned, and in Liao and Rao Vemuri (2002) the authors regarded the BSM audit data as documents and applied document classification terms: TF&IDF (term frequency and inverse document frequency) to weight features.

Most of above researches evaluated their approaches by the KDD CUP99 dataset. This means that their researches were designed for off-line detection and thus can't meet real-time demands for NIDSs. This is because the announced 41 features in KDD CUP99 were derived from connections, not packets. In fact, the 41 features presented in KDD CUP99 are complicated and varied (KDD CUP, 1999b; Middlemiss & Dick, 2003). The first 9 of 41 are intrinsic features which describe the basic features of individual TCP connections and can be obtained from raw TCPDUMP files; features 10–22 are content-based features obtained by examining the data portion of a connection and suggested by domain knowledge; features 23–31 are traffic-based features that are computed using a two-second time window ("time-based"), while features 32–41 are also traffic-based features but computed using a window of 100 connections ("host-based"). Moreover, the collection of attacks appearing in KDD CUP99 is out of date; for instance, in total only 12 DoS attacks appear in the KDD CUP99 dataset.

All features used in this paper are derived from packet headers and gathered using a two-second time window. The method proposed in this paper can be implemented to be real-time, i.e. making a decision every two seconds. If necessary, the time window can be reduced to one second or half a second. Basically, the result of KNN classification was adopted to design the fitness function in a genetic algorithm, to evolve the weight vectors of features. When the evolution was terminating, an optimal weight vector was obtained. Then, the least weighted features were dropped one by one. As one feature was being dropped, the proposed system was retrained and re-evaluated. Finally, the optimal weight vector with the best number of features and their weights were obtained.

The remainder of this paper is organized as follows: Section 2 briefly introduces genetic algorithm and KNN; Section 3 describes the proposed system; Section 4 discusses the experimental results; and Section 5 provides the conclusion.

2. Background

This section will briefly introduce the genetic algorithm and KNN, since the approach adopted in this paper is a GA/KNN hybrid.

2.1. Genetic algorithm (GA)

GA is essentially a type of search algorithm used to solve a wide variety of problems. Its goal is to create optimal solutions to problems (Holland, 1992). A potential solution is encoded as a sequence of bits, characters or numbers. This unit of encoding is called a gene, and the encoding sequence is known as a chromosome.

GA begins with a set of chromosomes, called a population, and an evaluation function which measures the fitness value of each chromosome. Usually, an initial population of chromosomes is created by complete randomization. During evolution, chromosomes are evaluated by the fitness function. Based on their fitness values, better chromosomes are selected as parents by selection procedure, and then the parents perform crossover and mutation to form new children chromosomes. Finally, some chromosomes in the current generation are replaced by the new ones, if necessary, to form the next generation. The evolution continues until some predefined situation is met, such as the number of iterations reached or an acceptable fitness value appearing. The iteration loop of a basic genetic algorithm is illustrated in Fig. 1.

The design of the fitness function is the most important part in GA because a good one can significantly improve the outcome of the GA. This study applied the classification result of KNN to design the fitness function.

2.2. KNN (k-nearest-neighbor)

One common classification scheme based on the use of distance measures is that of the k-nearest-neighbor. The KNN technique assumes that the entire sampling set includes not only the data in the set, but also the desired classification for each item. When a classification is to be made for a new item, its distance to each item in the sampling set must be computed. Only the k closest entries in the sampling set are considered further. The new item is then classified to the class that contains the most items from this set of k

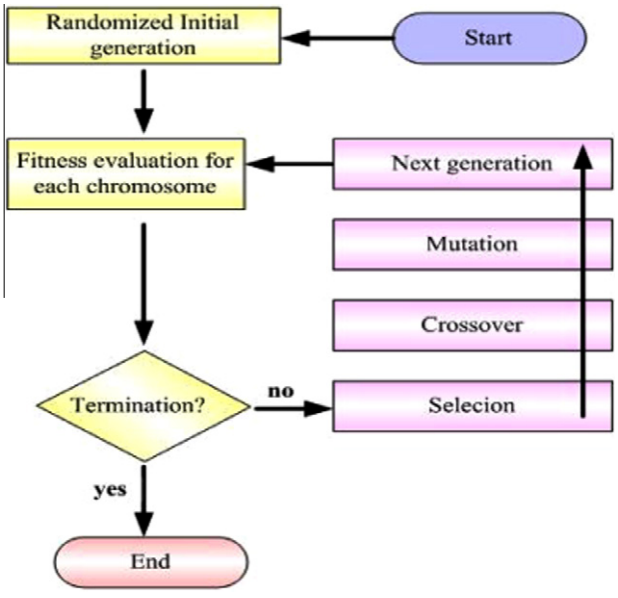

Fig. 1. Flowchart of GA.