Available online at www.sciencedirect.com

ScienceDirect

Pattern Recognition Letters 29 (2008) 918–924

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees

Cheng Xiang *, Png Chin Yong, Lim Swee Meng

Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

Received 8 May 2007; received in revised form 26 October 2007
Available online 26 January 2008

Communicated by N.S.V. Rao

Abstract

With increasing connectivity between computers, the need to keep networks secure progressively becomes more vital. Intrusion detection systems (IDS) have become an essential component of computer security to supplement existing defenses. This 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. Performance of this new approach is measured using the KDDCUP99 dataset and is shown to have high detection and low false alarm rates.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Bayesian clustering; Decision tree; False-negative; False-positive; Intrusion detection system (IDS)

1. Introduction

An increasing number of people are now connected to the web as PCs, PDAs and Internet access become more affordable. Current security measures such as firewalls, security policies, and encryption are not sufficient to prevent the compromise of private computers and networks. Therefore, intrusion detection systems (IDS) have become an essential component of computer security to supplement existing defenses. The infamous event of Yahoo being crippled for a few hours from a denial of service (DoS) attack in 2000, had raised lots of concern for network security and generated tremendous interests for designing better IDSs to protect the network.

As manual inspection of the audit trail data generated by operating systems is not feasible due to the incredibly large sizes of the data, data mining has been commonly used to automate the wading through audit data ever since the inception of the field (Denning, 1987; Lunt, 1988). For recent surveys, the reader is referred to Axelsson (2000) and Cabrera and Mehra (2002). Representative designs are Mukkamala et al. (2000) and Lee and Stolfo (2000). In particular, MADAM ID Lee and Stolfo (2000) (for Mining Audit Data for Automated Models for Intrusion Detection) is a good representative of the IDSs built with data mining tools, which has been considered by many researchers in this field as the bench-mark work for intrusion detection systems. However, there are two distinct weaknesses associated with MADAM ID. First of all, the detection rate for DoS attacks is relatively low, (79.9% for known attacks, and 24.3% for unknown attacks), which has to be substantially improved considering the fact that DoS is one of the most notorious and disruptive attacks. Secondly, the framework of MADAM ID is quite complicated, which includes programs for learning classifiers and meta-classifiers (Chan and Stolfo, 1993), association rules (Agrawal et al., 1993) for link analysis, and frequent episodes for sequence analysis (Mannila et al., 1995).

Several hybrid IDSs have been proposed recently to deal with the complexity of the intrusion detection problem by

* Corresponding author. Fax: +65 67791103.
E-mail address: elexc@nus.edu.sg (C. Xiang).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.01.008

combining different machine learning techniques. Pfahringer (2000), the winner of the KDDCUP99, fused 50×10 C5 decision trees using cost-sensitive bagged boosting algorithm, while Levin (2000) used a Kernel Miner to build an optimal decision forest. Both have very low false alarm rates but are unsatisfactory in detecting U2R and R2L attacks. Giacinto et al. (2003) used fusion of multiple classifiers and improved the detection rates for known attacks. However, the detection rates for unknown attacks were significantly lower than those of MADAM ID, which implies that the proposed method may over-fit the training data and not generalize well. Pan et al. (2003) took advantage of different classification abilities of neural networks and the C4.5 decision trees algorithm for different attacks. Depren et al. (2005) suggested a hybrid IDS consisting of an anomaly detection module, a misuse detection module and a decision support system. Zhang and Zulkernine (2006) combined the misuse detection and anomaly detection components in which the random forests algorithm was applied. Most recently, Hwang et al. (2007) proposed a hybrid system combining the advantages of low false-positive rate of signature-based intrusion detection system and the ability of anomaly detection system to detect novel unknown attacks, while Peddabachigari et al. (2007) advocated fusing decision trees and support vector machines as a hierarchical hybrid intelligent system model and an ensemble approach combining the base classifiers.

In order to further increase the intrusion detection rate, as well as to simplify the algorithm, a multiple-level tree classifier was recently proposed in (Xiang et al., 2004) to design an IDS, which contains three-levels of decision tree classification. It was shown to be easy to design and very efficient in detecting known attacks. However, a serious shortcoming of this approach is its high false alarm rate as well as low detection rate for unknown attacks. Thus as an improvement over this design, a new multiple-level hybrid classifier is proposed in this paper to reduce the false alarm rate to an industrially acceptable level while maintaining the low false-negative rate. While MADAM ID and other hybrid IDSs utilized a combination of data mining techniques, the design process of this new approach would be much simpler since only decision trees (Quinlan, 1993) and Bayesian clustering (Cheeseman et al., 1988) are involved.

The rest of the paper is organized as follows. Section 2 gives brief introductions on both decision trees (Quinlan, 1993) and Bayesian clustering (Cheeseman et al., 1988). The structure of the new multiple-level hybrid classifier is described in Section 3. And the experimental results are discussed in Section 4. Section 5 concludes the paper with suggestions for future work.

2. Preliminary

In this section, a brief introduction of the classification algorithms used in the hybrid IDS, i.e., the C4.5 algorithm for building decision trees and the Bayesian clustering algorithm, will be given.

2.1. C4.5 algorithm

The decision tree learning is one of the machine learning approaches for generating classification models. In this paper, C4.5, a later version of the ID3 algorithm (Quinlan, 1993), will be used to construct the decision trees for classification. In ID3, a decision tree is built where each internal node denotes a test on an attribute and each branch represents an outcome of the test. The leaf nodes represent classes or class distributions. The top-most node in a tree is the root node with the highest information gain. After the root node, one of the remaining attribute with the highest information gain is then chosen as the test for the next node. This process continues until all the attributes are compared or when all the samples are all of the same class or there are no remaining attributes on which the samples may be further partitioned.

The attribute with the highest information gain (or greatest entropy reduction) is chosen as the test attribute for the current node. Such an information-theoretic approach minimizes the expected number of tests needed to classify an object and guarantees that a simple (but not necessarily the simplest) tree is found.

Imagine selecting one case at random from a set S of cases and announcing that it belongs to some class C_j. The probability that an arbitrary sample belongs to class C_j is estimated by

$$p_i = \frac{\mathrm{freq}(C_j, S)}{|S|} \tag{1}$$

where $|S|$ denotes the number of samples in the set S. And so the information it conveys is $-\log_2 p_i$ bits.

Suppose we are given a probability distribution $P = \{p_1, p_2, \ldots, p_n\}$ then the information conveyed by this distribution, also called the entropy of P, is well known as

$$\mathrm{Info}(P) = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{2}$$

If we partition a set T of samples on the basis of the value of a non-categorical attribute X into sets $T_1, T_2, \ldots, T_m$, then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of T_i, i.e. the weighted average of $\mathrm{Info}(T_i)$

$$\mathrm{Info}(X, T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \times \mathrm{Info}(T_i) \tag{3}$$

The information gain, $\mathrm{Gain}(X, T)$, is then defined as

$$\mathrm{Gain}(X, T) = \mathrm{Info}(T) - \mathrm{Info}(X, T) \tag{4}$$

This represents the difference between the information needed to identify an element of T and the information needed to identify an element of T after the value of attribute X has been evaluated. Thus, it is the gain in information due to attribute X.

Remark 1. The notion of information gain introduced above tends to favor attributes that have a large number of values. For example, if we have an attribute D that has a distinct value for each record, then $\text{Info}(D, T)$ is 0, thus $\text{Gain}(D, T)$ is maximal. To compensate for this, a gain ratio criterion was introduced in (Quinlan, 1993).

2.2. Bayesian clustering

Bayesian clustering, which is also called AutoClass in (Cheeseman et al., 1988), is an unsupervised classification program that uses Bayesian inference to find the most probable classification given a description of the cases in the dataset.

AutoClass deals with the problem of automatic discovery of classes in data rather than the generation of class descriptions from labelled examples (so called supervised learning). In some sense, automatic classification aims at discovering the "natural" classes in the data. Although it is most suitable for problems where the training samples are not labelled at all by experts, it can also be applied to classification problems where the labeling information is available, as is the case of the intrusion detection problem for KDDCUP99, by simply ignoring the expert knowledge.

Consider the dataset $S = \{S_1, \ldots, S_I\}$, in which the samples, S_i, can be represented as vectors of attribute values, which can be described by discrete values such as "true" or "false", or integer values, or real numbers. Each observation of an instance is assumed to be generated by an underlying probabilistic model for the process, which refers to a probability distribution or density function (p.d.f.) that gives the probability of observing an instance possessing any particular attribute value vector.

Probabilistic models invariably contain free parameters, such as the Gaussian mean and variance, for computing instance probabilities. Thus it is useful to distinguish between the p.d.f.'s functional form and its parameter values, which are denoted by F and V, respectively.

For AutoClass, the fundamental model is the classical finite mixture distribution, which is a two component model. The first gives the interclass mixture probability that an instance S_i is a member of class C_j (where $j \in \{1, \ldots, n\}$ and n is the total number of classes), independently of any other information available. The interclass p.d.f. F_c is assumed to be a Bernoulli distribution characterized by the class number n and the probabilities of interclass parameters V_c. Each class C_j is then modeled by a class p.d.f. F_j giving the probability of observing the instance conditional on the assumption that instance S_i belongs to class C_j. The class p.d.f. F_j is a product of individual attribute p.d.f.'s F_{jk}; e.g. Bernoulli distributions for nominal attributes, Gaussian densities for real numbers, Poisson distributions for number counts, etc.

Given a set of data S there are two goals to accomplish: for any given classification p.d.f. F we seek the maximum posterior parameter values V, and irrespective of any V we seek the most probable F. Thus there are two levels of search. For any fixed probability functional form F specifying the number of classes n and their class models, we search the space of allowed parameter values for the maximally probable V. This is a real valued space of generally high dimension, subject to strong constraints between the parameters. There are many local maxima and there is no simple recipe to determine the global maximum except by trial and error. Thus parameter level search requires an expensive numerical optimization. The model level search involves the number of classes n and alternate class models. There are several levels of complexity. The basic level involves a single p.d.f. F_j common to all classes, with search over the number of classes, which is usually adopted in many applications. A second level allows the individual F_j to vary from class to class.

3. Multiple-level hybrid classifier

The network attacks fall into four main categories, DoS, Probe, U2R and R2L, which follow the standard classification of network intrusions as given in (Lippmann et al., 2000).

The overall structure of the multiple-level hybrid classifier is shown in Fig. 1. A model with four stages of classification is used for the hybrid classifier (see Fig. 1), whose structure takes a similar form as that of the three-level tree classifier (Xiang et al., 2004). The first stage of classification categorizes the test data into three categories (DoS, Probe, Others). U2R and R2L, and the Normal connections are classified as "Others" in this stage. The second stage splits "Others" into Attack and Normal categories. The third stage separates the Attack class from Stage 2 into U2R and R2L. The fourth stage further classifies the attacks into more specific attack types.

The main purpose at Stage 1 is to extract as many U2R, R2L and Normal connections as possible accurately from

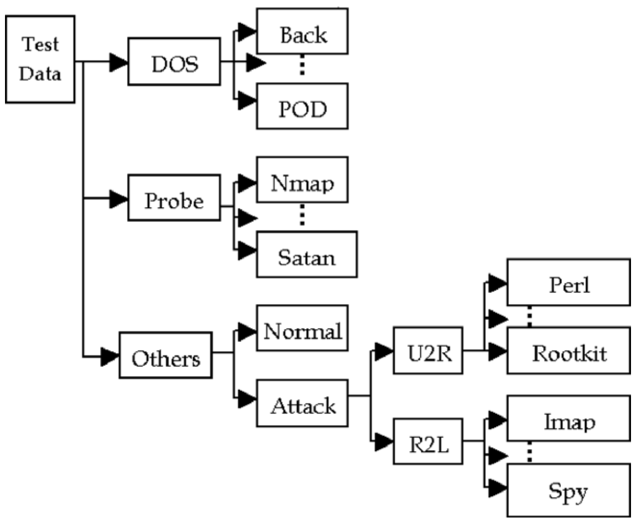

Fig. 1. Structure of the multiple-level hybrid classifier.

the data using C4.5 model. In the three-level tree classifier (Xiang et al., 2004), the data are split into Normal, DoS, Probe and "Others" (a class containing both U2R and R2L) at the first level. However, in the hybrid classifier as shown in Fig. 1, U2R, R2L and Normal connections are grouped together into one intermediate class, called "Others", in the first round. This is because U2R and R2L are generally quite similar with the normal connections. For example, an U2R attack is characterized by a process whereby any normal system user illegally gains access to the super-user privileges and a R2L attack occurs when a hacker who does not have an account on the victim machine, gains local access as a user of the victim machine by sending network packets through standard protocols. In both cases, after the hacker gains the illegal access of the victim's computer, the connection record generated will be very similar to a normal user connection record. This makes it very hard to distinguish between U2R, R2L and Normal. It would be difficult for the automatic feature construction process, which is based on frequent patterns of connection records, to produce any statistical feature for these attacks, which might be the main reason for the high false alarm rate produced by the three-level tree classifier (Xiang et al., 2004). In order to decrease the false alarm rate, it was decided to combine these three types of connections together and classify them as "Others" which will be further separated by lower-level classifier.

DoS and Probe attacks connections on the other hand are generally more different. A DoS attack is one in which the attacker attempts to paralyze the network making it incapable of handling legitimate requests, or denies legitimate users access to a machine. They often contain special information like request for connection and they occur in large numbers. Probe on the other hand, scan a network of computers to gather information or find known vulnerabilities.

Stage 2 is essentially to separate out the U2R, R2L and Normal connections, which is the critical step where the Bayesian clustering (AutoClass) is involved, which is also why the proposed IDS is coined hybrid classifier: both supervised learning (decision tree construction) and unsupervised learning (clustering) are involved. In fact, decision tree was initially implemented for this stage, but the classification result was far from satisfactory. Then it was decided to utilize the clustering algorithm, which surprisingly resulted in much better recognition performance as will be shown later in Section 4. The clusters are first generated by AutoClass from the training data of U2R, R2L and Normal connections. Then these clusters will be labelled by the rule that they would be categorized as Normal connections only if none of the training samples within the clusters are U2R or R2L attacks. In other words, the clusters which contain at least one U2R or R2L attack are all labelled "Attack". Other clustering algorithms such as *K*-means are also experimented with and the choice of AutoClass is made through comparison studies.

The training data used in Stage 2 is 10,000 randomly selected Normal connections and all of the available U2R and R2L attacks in the original training data for KDD-CUP99. The number of 10,000 is chosen as a tradeoff between classification accuracy and convergence speed of the AutoClass algorithm. Therefore a total of 11,178 connections are used as the training data for Bayesian clustering.

Not all of the 41 attributes are used for clustering. Only four attributes are selected after weighting the significance of each attribute using a combination of criteria such as information gain. The four attributes selected are duration, services, src_bytes and dst_bytes. In addition, in the search parameter specification of AutoClass, the default value of "REL_DELTA_RANGE" is decreased from 0.0025 to 0.0005 to tighten the convergence range of the clustering so that each cluster is more likely to contain connections of similar type. Also, AutoClass begins each clustering attempt with a certain number of clusters and this number may evolve with the Bayesian clustering algorithm as described in Section 2.2. In our experiment, the initial number of clusters is fixed at 250 after some trial and error. The rationale is that with greater number of clusters, each connection will more likely be singled to a cluster with similar types. After the AutoClass algorithm converges, the training data is separated into 178 clusters (which is smaller than the initial number of 250) and 31 of them are labelled "attack" clusters.

In Stage 3, decision trees are used to separate out the U2R and R2L. This step is made easier due to the fact that the Normal connections have been filtered out beforehand. Furthermore, in order to improve the accuracy and shorten the processing time of the classification process, feature selection is first performed to remove all the redundant attributes. These redundant attributes may contain false correlations which will hinder the process of detecting intrusions. Some of these attributes may also be unnecessary as they provide duplicated information. Removing these attributes reduces the complexity of the classification tree produced, reduces processing time and creates a more generalized classification. For this stage only 14 of the 41 attributes, which have the highest information gain, are used in building the decision tree by C4.5 algorithm. The 14 attributes selected are: src_bytes, dst_bytes, dst_host_srv_count, root_shell, num_compromised, dst_host_srv_diff_host_rate, hot, num_file_creations, dst_host_same_srv_rate, dst_host_count, is_guest_login, num_root, duration, service.

The last stage further classifies all the various class of attacks into more specific attack types based on the given training data using the C4.5 decision trees. It tells us what attacks were being employed to invade the system and helps the system administrator understand them in more details. This classification is only effective for known attacks as it requires that particular type of labelled training data to be present.

4. Experiments

In this section, the data sets and software used in the experiment are described, and the classification results using the proposed multiple-level hybrid classifier are presented and compared with the detection results from the three-level tree classifier as well as those of the winners of KDDCUP99 competition.

4.1. Training and test data

The dataset used in the experiment is taken from the Third International Knowledge Discovery and Data Mining Tools Competition (KDDCUP99) KDD Cup (1999). Each connection record is described by 41 attributes. The list of attributes consists of both real-valued and discrete type variables, with statistical distributions varying drastically from each other, which makes the intrusion detection a very challenging task.

Table 1 shows the training data used and the test data. The training data set contains 22 attack types and is distributed among five million records. The test data has the same 22 attack types as well as 17 more additional attack types, e.g. mailbomb, mscan, and snmpgetattack etc., distributing among 311,029 records. These are the "unknown" attacks and will test the model on its capability to detect new unknown variant attacks.

Remark 2. It has been observed that the class distribution in training data will affect classifier learning significantly since the decision trees are built with the statistical information of the samples, and naturally occurring class distribution may not be the best in terms of classification performance. Thus, not all the connection records from original training set are used for building the decision tree in Stage 1 as shown in Table 1. Some heuristics have been employed in selecting the training set. Basically, all the samples will be used if the number of samples is relatively small, and only a small subset will be randomly selected from the original training data if the size of the training data is huge. For example, for DoS, a subset of records are selected from smurf and neptune attacks while the rest of the DoS training attacks are fully included.

4.2. Software

Weka 3: Data Mining Software in Java, a collection of machine learning programs for data mining tasks written in Java, is chosen to build the tree classifiers in the experiment. The version used is Weka 3.4. The J4.8 algorithm, which is Weka's implementation of the C4.5 decision tree learner, is utilized.

AutoClass is an unsupervised Bayesian classification (Cheeseman et al., 1988) system that seeks a maximum posterior probability classification as discussed in Section 2.2. The current version used in this project is AutoClass C, which is written in C.

4.3. Experiment result

The overall detection rates for the five categories are shown in Table 2 and they are compared to those from winning entry and runner up of KDDCUP99 and three-level classification as shown in Table 3. And the overall false-negative and false-positive rates are shown in Table 4. The proposed hybrid classifier has achieved the highest detection rates for all attack types as shown in Table 3.

All the models are able to detect DoS and Probe attacks with high percentage and this is expected as these attacks are frequent episodes attacks. In particular, the detection rates of both known and unknown DoS attacks obtained by the proposed hybrid model are 99.19% and 80.63%, respectively, which are substantially higher than those from the MADAM ID (79.9% and 24.3% for known and unknown attacks). The improvement of detecting Probe (93.40%) over the winners of KDDCUP99 (83.30% and 84.52%) is also significant.

Table 1
Connection records of training and test sets

Type of connection	Available training set	Training set	Test set
DoS	3883370	8591	229853
Probe	41102	8218	4166
U2R	52	52	70
R2L	1126	1126	16347
Normal	972780	3000	60593

Table 2
Detection rates of hybrid classifier (%)

Category	Known attacks	Unknown attacks
DoS	99.19	80.63
Probe	99.71	85.02
U2R	66.67	77.42
R2L	89.14	22.56
Normal	96.80	–

Table 3
Comparison of overall detection rates (%)

Category	Bagged boosted C5 trees	Kernel miner	Three-level tree classifier	Hybrid classifier
DoS	97.10	97.47	97.35	98.66
Probe	83.30	84.52	93.23	93.40
U2R	13.20	11.84	61.43	71.43
R2L	8.40	7.32	23.69	46.97
Normal	99.50	99.42	42.73	96.80

Table 4
Overall normal-intrusion detection rates (%)

	Predicted normal	Predicted intrusion
Actual normal (60,593 total)	58657	1936(3.2%)
Actual intrusion (250,436 total)	8087 (3.23%)	242349

Both the three-level classifier and the hybrid classifier perform significantly better in detecting U2R and R2L attacks. However, the high detection rate for the three-level tree classifier comes at the price of high false alarm rate as well. This is where the hybrid classifier outperforms the three-level tree classifier. It is shown in Table 4 that the false alarm rate is merely 3.2% which is an astonishing improvement over that of the three-level tree classifier (57%). The false alarm rate is slightly more than those of the winners of KDDCUP99 and it may be argued that this can be significant as normal connections are at orders of magnitude greater than attack connections. However the impact of a breached network for many businesses and government agencies might be significantly greater and U2R and R2L attacks are likely to cause more damages than DoS or Probe attacks. Hence with much higher detection rates for U2R and R2L attacks as well as better overall detection rates, it may be concluded that the proposed hybrid classifier outperforms the rest of the models.

The reason why the R2L detection rate is still very low at 46.97% is mainly due to the misclassification of one particular type of the R2L connections, Snmpgetattack, found only in the test data. Snmpgetattack makes up nearly half of the test data for R2L (7741/16347) and is not present in the training data. All of them have been wrongly classified as Normal connections. The reason for this is actually due to the fact that in the DARPA98 where KDDCUP99 data were obtained from, the SNMP community password is set to the default ("public") and is never changed. There were Snmpguess attacks where SNMP requests were sent to the internal router of the SNMP community guessing the password until a correct response is received from the router. Once the correct password is being obtained by the hacker, he/she is then able to monitor the router without being detected using the guessed password. The traffic generated will be very similar to that of a normal connection, and sometimes they can be exactly the same as found in the test data. Among the 41 attributes used to describe a connection in the KDDCUP99 there is no attribute which is able to distinguish a Snmpgetattack from a normal connection. Attributes which contain information like password type are not present in the data of KDDCUP99, so the transformation technique used to transform data from DARPA98 (Lippmann et al., 2000) to KDDCUP99 is not appropriate as observed by Bouzida and Cuppens (2006).

Speed is always a concern as the purpose of intrusion detection system is for on-line applications. The most time-consuming part of course comes from trying to find the ideal hybrid classifier model, the correct combination of parameters, better ways of training the data and the search for a proper training data set etc. Assuming the structure of the classifier and the search parameters of the algorithms are finalized, the running time for the training and testing of each stage is recorded and listed in Table 5. The experiments are conducted using a Pentium-4, Dell PC, with CPU speed of 1.86 GHz and RAM of 512 MB. Obviously the training takes much longer time than testing as shown in Table 5. In particular, the Bayesian clustering takes around one day (24 h) to converge, which is much more computational expensive than building decision trees. However, once the decision trees are constructed and the clusters are obtained and labelled, it is very fast to classify each connection record. The total running time for classifying all the connection records, i.e. testing time, is only 257 s. As there are 311,029 connection records in the test set, the average time to classify each connection record is merely $257/311029 = 0.0008$ s, which is fast enough for any on-line detection.

Table 5
Running time of hybrid classifier

Stages	Training (s)	Testing (s)
1	64	44
2	86423	83
3	3	17
4	131	113

5. Conclusion

A multiple-level hybrid classification model combining decision trees and Bayesian clustering has been proposed in this paper. Experimental results with this new scheme on KDDCUP99 data set were compared with other popular approaches such as multiple-level tree classifier and winners of KDDCUP99. It was shown that this new approach is very efficient in detecting intrusions with an extremely low false-negative rate of 3.23%, while keeping an acceptable level of false-alarm rate of 3.2%. Although the false alarm rate was reported to be as low as 0.5% for the KDDCUP99 winner (Pfahringer, 2000), the corresponding false negative rate was merely 9.1%, which is much higher than our result (3.23%). It is our belief that keeping false-negative rate as low as possible while maintaining an acceptable level of false alarm rate is essential for IDS, since the false alarm might bring inconvenience to the administrators of the intrusion detection system but it is more critical to ensure that intrusions are not misclassified as normal connections, which otherwise might cause greater damages to the network.

Compared to other hybrid IDSs proposed in the literature, the major novel feature of this approach is the combination of both supervised learning (tree classifier design) and unsupervised learning (clustering analysis). Most of other IDSs have utilized various techniques related to supervised learning such as decision trees, neural networks, and SVMs etc. since the training data are clearly labelled, as is the case of KDDCUP99 Data. The unsupervised learning would be employed normally when the labels of the training data are unknown, which may explain the fact that clustering analysis has been very little used in the design of IDS for KDDCUP99 Data. However, it has been demonstrated in this paper that the clustering analysis (unsupervised learning) may be critical for improving the detection rate even when the training data are plainly labelled. Although recently there are a few reports (Julisch, 2003; Zanero and Savaresi, 2004; Petrovic et al., 2006) on applying unsupervised learning to IDS design, the integration of both supervised and unsupervised learning has rarely been used.

Although the proposed multiple-level classification IDS looks promising, there is still a large room to improve the detection rates for unknown attacks. Better anomaly detection methods may be used to improve this model as anomaly detection stands a better chance of detecting unknown attacks.

More importantly, there remains the intriguing issue regarding the general utility of this approach of combining clustering with supervised learning. Is it possible to apply this scheme to other types of pattern recognition problems? Can we develop a general framework for this approach? For instance, is it always a good idea to use clustering first to explore the natural structure of the data, then followed by further refinement through supervised learning? Work is currently under progress to investigate all these open questions.

Acknowledgements

The research reported here was supported by NUS Academic Research Fund R-263-000-362-112. The authors would also like to thank the associate editor and the anonymous reviewers for their critical comments to improve the quality of this paper.

References

Agrawal, R., Imielinski, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: Proc. 1993 ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD'93, Washington, DC, May 26–28), pp. 207–216.

AutoClass C – General Information. <http://ic.arc.nasa.gov/ic/projects/bayes-group/AutoClass/AutoClass-c-program.html>.

Axelsson, S., 2000. Intrusion detection systems: A taxonomy and survey, Technical Report 99-14, Dept. of Computer Engineering, Chalmers University of Technology, Sweden.

Bouzida, Y., Cuppens, F., 2006. Detecting known and novel network intrusions. In: Proc. IFIP TC-11 21st Internat. Information Security Conf. (SEC 2006), pp. 258–270.

Cabrera, J.B.D., Mehra, R.K., 2002. Control and estimation methods in information assurance – a tutorial on intrusion detection systems. In: Proc. 41st IEEE Conf. on Decision and Control, pp. 1402–1407.

Chan, P.K., Stolfo, S.J., 1993. Toward parallel and distributed learning by meta-learning. In: Proc. AAAI Workshop on Knowledge Discovery in Database, pp. 227–240.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D., 1988. AutoClass: A Bayesian classification system. In: Proc. Fifth Internat. Conf. on Machine Learning.

Denning, D., 1987. An intrusion detection model. IEEE Trans. Software Eng. 13 (2), 222–232.

Depren, O., Topllar, M., Anarim, E., Ciliz, M.K., 2005. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Syst. Appl. 29, 713–722.

Giacinto, G., Roli, F., Didaci, L., 2003. Fusion of multiple classifiers for intrusion detection in computer networks. Pattern Recognition Lett. 24, 1795–1803.

Hwang, K., Cai, M., Chen, Y., Qin, M., 2007. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. IEEE Trans. Depend. Secure Comput. 4, 41–55.

Julisch, K., 2003. Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inform. Syst. Security 6, 443–471.

KDD Cup, 1999. Data, Information and Computer Science, University of California, Irvine. <http://KDD.ics.uci.edu/databases/KDDcup99/KDDcup99.html>.

Lee, W., Stolfo, S.J., 2000. A framework for constructing features and models for intrusion detection systems. ACM Trans. Inform. Syst. Security 3 (4), 227–261.

Levin, I., 2000. KDD-99 classifier learning contest LLSoft's results overview. SIGKDD Explor. ACM SIGKDD.

Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A., 2000. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In: Proc. 2000 DARPA Inform. Survivability Conference and Exposition (DISCEX), vol. 2. IEEE Computer Society Press, Los Alamitos, CA, pp. 12–26.

Lunt, T., 1988. Automated audit trail analysis and intrusion detection: A survey. In: Proc. 11th National Computer Security Conference, pp. 65–73.

Mannilla, H., Toivonen, H., Verkamo, A.I., 1995. Discovering frequent episodes in sequences. In: Proc. 1st Internat. Conf. on Knowledge Discovery in Databases and Data Mining.

Mukkamala, R.K., Gagnon, J., Jajodia, S., 2000. Integrated data mining techniques with intrusion detection. In: Atluri, V., Hale, J. (Eds.), Research Advances in Database and Information Systems Security. Kluwer Publisher, pp. 33–46.

Pan, Z.S., Chen, S.C., Hu, G.B., Zhang, D.Q., 2003. Hybrid neural network and C4.5 for misuse detection. In: Proc. 2003 Internat. Conf. on Machine Learning and Cybernetics, vol. 4, pp. 2463–2467.

Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J., 2007. Modelling intrusion detection system using hybrid systems. J. Network Comput. Appl. 30, 114–132.

Petrovic, S., Alvarez, G., Orfila, A., Carbo, J., 2006. Labelling clusters in an intrusion detection system using a combination of clustering evaluation techniques. In: Proc. 39th Annual Hawaii Internat. Conf. on System Sciences, pp. 129b–129b.

Pfahringer, B., 2000. Winning the KDD99 classification cup: Bagged boosting. SIGKDD Explor. 1 (2), 67–75.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kauffman.

Weka 3: Data Mining Software in Java, University of Waikato, New Zealand. <http://www.cs.waikato.ac.nz/ml/weka>.

Xiang, C., Chong, M.Y., Zhu, H.L., 2004. Design of multiple-level tree classifiers for intrusion detection system. In: Proc. 2004 IEEE Conf. on Cybernetics and Intelligent Systems, December, Singapore, pp. 872–877.

Zanero, S., Savaresi, S.M., 2004. Unsupervised learning techniques for an intrusion detection system. In: Proc. 2004 ACM Symposium on Applied Computing, pp. 412–419.

Zhang, J., Zulkernine, M., 2006. A hybrid network intrusion detection technique using random forests. In: Proc. 1st Internat. Conf. on Availability, Reliability and Security, ARES 2006, Vienna, Austria, pp. 262–269.