Distributed Detection of Network Intrusions Based on a Parametric Model

Yan-guo Wang, Xi Li, and Weiming Hu
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
Beijing, China
{ygwang,lixi,wmhu}@nlpr.ia.ac.cn

***Abstract*—With the increasing requirements of fast response and privacy protection, how to detect network intrusions in a distributed architecture becomes a hot research area in the development of modern information security systems. However, it is a challenge to build such a system, given the difficulties brought by the mixed-attribute property of network connection data and the constraints on network communication. In this paper, we present a framework for distributed detection of network intrusions based on a parametric model. The parametric model can explicitly reflect the distributions of different intrusion types and handle the mixed-attribute data naturally. Based on the model, we can generate an accurate global intrusion detector with a very low cost of communication among the distributed detection sites, and no sharing of original network data is needed. Experimental results demonstrate the advantages of the proposed framework in the distributed intrusion detection application.**

I. Introduction

With the great damage caused by network intrusions, information security becomes more and more important in the development of network-based information systems, such as e-business and e-government. Generally, an information security system contains the following components:

1) Protection: Intrusion preventive measures, such as user authentication and encryption.

2) Detection: Try to detect security violating behaviors as soon as they occur.

3) Reaction: Reactions to the security violations detected, such as automatic alarm of intrusions, or cut the specific network connections where the attacks are from.

4) Recovery: Repair the damage on the information system caused by the intrusions.

Intrusion detection is a crucial module of the information security systems, as the subsequent reaction and recovery modules can take effect only if the intrusions are correctly detected in time. Thus more and more attention has been paid to the development of effective intrusion detection techniques.

A. State of the Art

Statistical methods were first introduced into the development of intrusion detection algorithms. Denning [1] proposes a detection method, where statistical profiles for normal behaviors are constructed and applied to detect anomalous behaviors as intrusions. Li and Manikopoulos [2] extract several representative parameters of network flow, and these parameters are modeled with a hyperbolic distribution. Peng et al. [3] use a nonparametric cumulative sum algorithm to analyze the statistics of network data, and further detect anomalies on the network.

Data mining techniques are also widely used in intrusion detection. Lee et al. [4] introduce the concepts of association rules and frequent episodes to describe the characteristics of normal network activities. The frequent episodes that are of little importance are dynamically discarded in [5] during the construction of network behavior models. Otey el al. [6] propose a general-purpose outlier detection algorithm based on the mining of frequent itemsets, and the algorithm is extended to fit the dynamic and streaming data environments.

Recently, many machine learning methods are introduced into the intrusion detection research. For supervised learning, Bonifacio et al. [7] use neural networks to distinguish between intrusions and normal network activities. Later, Hierarchical Neural Networks [8] and Evolutionary Neural Networks [9] are applied to the analysis of intrusion patterns. Another algorithm with great generalization ability, Support Vector Machines (SVMs), are also exploited in [10], [11].

As for unsupervised learning, Xian et al. [12] combine the fuzzy K-means with clonal selection algorithm in the detection of intrusions. An incremental version of K-means algorithm is used in [13]. Another popular clustering tool, self-organizing map (SOM), also attracts great attention [14], [15]. Sarasamma et al. [16] propose a Hierarchical SOM to select features that are used in different layers of SOM. There are also many other pieces of work on this research area [17], [18].

B. Challenges for Distributed Intrusion Detection

With the increasing requirements of fast response to intrusions and protection of data privacy, it becomes a new trend in the intrusion detection research to develop intrusion detection techniques that are implemented in distributed architectures [19]. In the distributed environments, the detection of intrusions is performed on local detection sites, which are multiple distributed computing nodes near the data sources. Compared with the traditional centralized detection ways in which all the network data are sent to a central site to detect intrusions, the response to intrusions is faster in the distributed

detection architectures, and the original network data from each distributed data source are no longer shared outside.

However, due to the limitation on the amount of observation data, it is infeasible to construct an accurate detection model using only local data, so the detection ability of each distributed site is greatly restricted. Sharing of local models is helpful to the improvement of detection performance, but what information to share and how to share is a challenge to the distributed intrusion detection systems, since the volume of communication among the distributed detection sites should be minimized, and the sharing of original network data should be avoided for the consideration of privacy protection. Furthermore, the variety of attributes also brings difficulties to local information sharing. There are various types of attributes for network data, including both categorical and continuous ones, and the ranges of value for different attributes differ greatly, from $[0, 1]$ to $[0, 10^7]$. It is a difficult task to extract a concise sharing model from these mixed-attribute data, which can effectively improve the detection ability on each distributed detection site. Recently, Otey et al. [6] use frequent itemsets to detect mixed-attribute outliers in the distributed mining environments. Although local information can be shared in the algorithm, the communication cost is very high. Furthermore, many original network data need to shared among the distributed sites, which is not suitable for the specific application.

C. Our Approach

In this paper, a framework for distributed detection of network intrusions is proposed, which is based on a parametric model. The parametric model is a hybrid of Adaboost ensemble [20] and Gaussian Mixture Model (GMM), which results in a concise sharing model with useful information of each intrusion type. Based on the parametric sharing models from the distributed detection sites, a global intrusion detector is finally constructed in a heuristic approach. The advantages of our method are as follows:

- In the Adaboost ensemble classifier, the weak classifiers are constructed on each individual feature dimension, both for continuous and categorical ones, so that the difficulties of mixed-attribute data from the variety and the complicated relationships among these features are successfully handled.
- The weak classifiers are based on the GMM of each intrusion type, then a parametric model is generated which is suitable for sharing, since it gives a good description of the distribution for each type of intrusions.
- Since only the concise parametric models are shared among the distributed detection sites, the volume of communication is very little, which greatly reduces the network bandwidth occupation and speeds up the response to intrusions.
- Due to the great descriptive power of the parametric model, the final heuristically constructed global detector reaches a considerable improvement on the detection ability of each distributed detection site, and no private network data need to be shared for the detection.

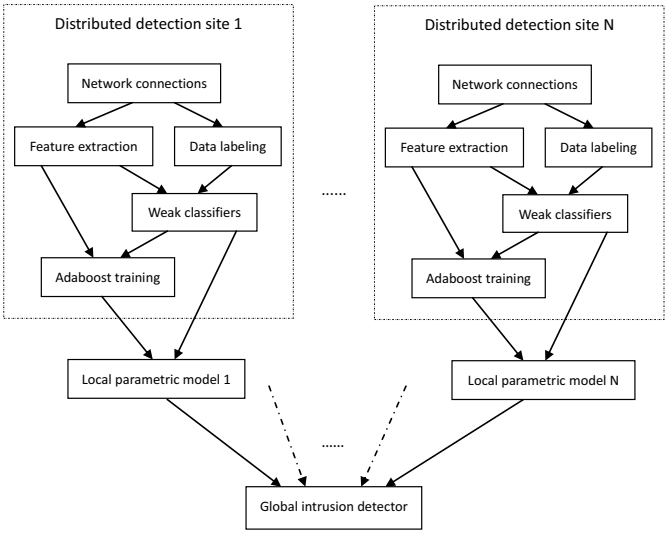

Fig. 1. Framework of Our System

The rest of paper is organized as follows. Section II gives an overview of our framework. Section III describes the methodology of our distributed detection algorithm in detail. Section IV reports some experimental results which show the advantages of the proposed method. Then we draw the conclusion in the last section.

II. System Overview

The framework of our distributed detection method for network intrusions is illustrated in Fig. 1. There are mainly six modules for the system.

A. Feature Extraction

For each network connection, three groups of features are extracted for detecting intrusions [21].

- basic features of individual Transmission Control Protocol (TCP) connections;
- content features within a connection suggested by domain knowledge;
- traffic features computed using a two-second time window.

The above features are commonly used in intrusion detection. The framework for constructing these features can be found in [22].

B. Data Labeling

Since supervised learning techniques are applied in the system, a set of training data need to be labeled before the training phased. There are many types of intrusions on the Internet, which differ greatly from each other on the characteristics of the connection data. Generally, the main objective of intrusion detection is to make a decision whether

a network connection is an intrusion, while the intrusion's specific type is of little concern. However, in order to extract a description for the distribution of each intrusion type, which is helpful for the information sharing among the distributed detection sites, we labeled the intrusion data as "-1", "-2", ... for different intrusion types, and the normal connection data are all labeled as "1".

C. Design of Weak Classifiers

The weak classifiers in our system are the ones on each individual feature dimension, so that we can handle the mixed-attribute data of network connections naturally, and make full use of information on all the features to get better intrusion models. As a simple and effective tool, GMM is exploited in the design of weak classifiers to generate an explicit distribution of each intrusion type.

D. Adaboost Training

Adaboost training is implemented on each distributed detection site using only the local training data, then a set of ensemble weights are produced at the end of training, which reflect the importance of each feature for the detecting of intrusions.

E. Generation of Local Parametric Models

Based on the weak classifiers and ensemble weights obtained from the former two modules, a concise but effective local parametric model for each distributed detection site is achieved naturally.

F. Construction of Global Detector

Finally, after the local parametric models from the distributed detection sites are shared, we construct a global intrusion detector, which is then used to detect intrusions on each distributed site.

III. Methodology

In this section, we describe the detailed techniques of our method, followed by some discussions.

A. Weak Classifiers

For each network connection, the feature values extracted form a vector

$$\mathrm{x} = [x_1, x_2, ..., x_D], \tag{1}$$

where D is the number of features extracted. Note that in the D dimensions, there are both continuous and categorical features, and the ranges of value on these features differ greatly from each other. Although they contain important information for the data distribution, the categorical features are inappropriately modeled or simply discarded by most of existing algorithms, which restricts the detection ability of the resulted intrusion detectors. Moreover, due to the demand of fast response in the security domain, the detection models with high complexity are not fit for the specific application. Recently, Otey et al. [6] use frequent itemsets to capture the complicated dependencies among the mixed-attribute data, but there are a large amount of statistics to be shared about the frequent itemsets, which means a high communication cost in the distributed detection systems.

In our previous work [23], the mechanism of ensemble classifier is applied, and the weak classifiers are constructed on each individual feature dimension, so that the difficulties from the variety and complex relationships among these features are well handled. However, the weak classifiers used in [23] are decision stumps. Although work well on a single detection site, ensemble detectors based on decision stumps make it inconvenient to share information among the distributed sites.

Here we introduce the GMM-based weak classifier on each individual feature dimension. Suppose the GMM of data class c on the j-th feature is

$$\Theta_j^c = \{\omega_i, \mu_i, \sigma_i\}_{i=1}^K, \tag{2}$$

where $j \in [1, D]$, $c = 1, -1, -2, ...$ is the class label as described in section II.B, K is the number of Gaussian components, then the weak classifier on the j-th feature is constructed as follows:

$$h_j(\mathrm{x}) = \arg\max_c \{P(\mathrm{x}|\Theta_j^1), \frac{1}{M}P(\mathrm{x}|\Theta_j^{-1}), \frac{1}{M}P(\mathrm{x}|\Theta_j^{-2}), ...\}, \tag{3}$$

where M is the number of intrusion types appearing on the local detection site. The involving of parameter M is of great importance to reduce the false alarm rate for the final ensemble classifier. Note that the parameters of Θ_j^c can be easily obtained from the training data by K-means clustering or Expectation-Maximization (EM) learning [24].

B. Adaboost Training

Adaboost [20] is one of the most popular machine learning algorithms developed in recent years, which has shown great power in many applications. Let $\{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_N, y_N)\}$ be the local training data on a distributed detection site, where N is the number of training data. Based on the weak classifiers (3) described above, we present our Adaboost-based training of local intrusion detectors as below:

1) Initialization:

$$w_n^{(1)} = 1/N, \quad n = 1, ..., N.$$

2) Selection of Weak Classifiers (for $t = 1, ..., T$):

Let $\varepsilon_j^{(t)}$ be the sum of weighted classification error for the weak classifier h_j in the t-th iteration

$$\varepsilon_j^{(t)} = \sum_{n=1}^{N} w_n^{(t)} \cdot I[\mathrm{sign}(y_n) \neq \mathrm{sign}(h_j(\mathrm{x}_n))], \tag{4}$$

where

$$I[\gamma] = \begin{cases} 1 & \text{if } \gamma \text{ is true} \\ 0 & \text{if } \gamma \text{ is false} \end{cases}. \tag{5}$$

Choose the weak classifier $h^{(t)}$ with the minimum of weighted classification error

$$h^{(t)} = \arg\min_{h_j} \varepsilon_j^{(t)}. \tag{6}$$

Suppose $\varepsilon^{(t)}$ is the related weighted classification error of $h^{(t)}$, then we update the weight of each training data according to the following rules,

$$w_n^{(t+1)} = w_n^{(t)} \times \begin{cases} \frac{1}{2(1-\varepsilon^{(t)})} & \text{if } \text{sign}(h^{(t)}(\mathrm{x}_n)) = \text{sign}(y_n) \\ \frac{1}{2\varepsilon^{(t)}} & \text{if } \text{sign}(h^{(t)}(\mathrm{x}_n)) \neq \text{sign}(y_n) \end{cases}. \tag{7}$$

3) Strong ensemble classifier:

The final strong classifier generated by Adaboost training is

$$H(\mathrm{x}) = \text{sign}(\Sigma_{t=1}^{T} \text{sign}(h^{(t)}(\mathrm{x})) \cdot \alpha^{(t)}), \tag{8}$$

where $\alpha^{(t)} = \lg \frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}$, which reflects the importance of the feature dimension related to the weak classifier $h^{(t)}$.

The final strong classifier in Adaboost is a linear weighted sum of the weak classifiers, and the voting weights are derived from the weighted classification errors of these weak classifiers, which are learned on the evolving sampling distribution of the training data set. The evolving weight $w_n^{(t)}$ plays a key role in Adaboost. It indicates the importance of the n-th training sample while generating the t-th weak classifier. The weights of samples that are wrongly classified by the current weak classifier are increased, and the others are decreased, so that more attention is paid to the samples that are difficult to classify while generating the next weak classifier. Theoretical proof has been given in [20], which implies the convergence of weighted classification error for the final strong classifier:

$$\sum_{n=1}^{N} w_n^{(1)} \cdot I[H(\mathrm{x}_n) \neq \text{sign}(y_n)] \to 0, \quad \text{as } T \to \infty. \tag{9}$$

C. Local Parametric Models

Subsequent to the above two modules of weak classifier construction and Adaboost training, a local parametric model is formed naturally for each distributed detection site:

$$\Psi = \{\Psi_w, \Psi_d\}, \tag{10}$$

where

$$\Psi_w = \{\alpha^{(t)} | t = 1, ..., T\} \tag{11}$$

is a set of ensemble weights from (8), and

$$\Psi_d = \{\Theta_j^c | j = 1, ..., D; c = 1, -1, -2, ...\} \tag{12}$$

is a set of GMM parameters from (2).

Compared with the distributed outlier detection algorithm proposed by Otey et al. [6], where a large amount of statistics about frequent itemsets need to be shared among the distributed detection sites, the parametric sharing model (10)∼(12) is not only concise to be suitable for information sharing, but also capture the distribution of the mixed-attribute data from each intrusion type very well, which is very useful to generate an accurate intrusion detection model.

D. Global Intrusion Detector

After being shared among the distributed detection sites, the local parametric sharing models (10)∼(12) should be combined into a global intrusion detector, which can utilize the information from all the distributed detection sites to generate an more accurate detection model. Since the training data on each distributed site are very limited, which may be adequate only for some specific intrusion types that appear much on a distributed site, we construct the global intrusion detector in the following simple manner:

$$G(\mathrm{x}) = \begin{cases} -1 & \text{if there exists at least one } H(\mathrm{x}) = -1 \\ 1 & \text{else} \end{cases}. \tag{13}$$

The idea behind (13) is intuitive: if at least one distributed detector has enough confidence that a connection data x is an intrusion, then it is very likely that x is actually a network intrusion. Though it is a little simple, the global detector (13) works well in the distributed intrusion detection application.

E. Discussions

In the intrusion detection, it usually needs a large amount of labeled connection data for training. In order to reduce the human workload of manual labeling, we proposed a hierarchical graph-theoretic clustering active learning algorithm in [25], which can automatically select a small part of highly informative data for human labeling while maintaining favorable accuracy in intrusion detection.

As a special characteristic of information security domain, there are many new intrusion types produced on the Internet, and the ability of detecting the new types of intrusion should be incorporated into the current detector soon after their emergence. To this end, the idea of online boosting [26] can be applied into our system in order to avoid the frequent retraining processes and adapt quickly to the changing network environments. The online version of EM learning algorithm [27] can also be used to further speed up the construction of GMM in the framework, which has been successfully utilized in many applications [28], [29].

IV. Experiments

The Knowledge Discovery and Data Mining Cup 1999 data set [21] is used to test our algorithm, which is a widely used benchmark data set for the evaluation of network intrusion detection methods. It was developed for the intrusion detection evaluation program by 1998 Defense Advance Research Projects Agency, which was prepared and managed by the Lincoln Laboratory, Massachusetts Institute of Technology. A network environment was set up to simulate a typical U.S. Air Force LAN, where a wide variety of intrusions were simulated as in a real military network. Nine weeks of TCP/IP connection data were collected, and they were labeled for testing intrusion detection algorithms. For each network connection, 41 features are extracted, including 9 categorical and 32 continuous features. The training data set contains 97278 normal connections and 396743 network intrusions, and the test data set contains 60593 normal connections and

TABLE I
RESULTS WITH THREE DISTRIBUTED DETECTION SITES

Distributed Detection Site	Detection Rate using Local Model	Detection Rate using Global Model
Site I	0.2528	0.8914
Site II	0.2737	0.9171
Site III	0.7544	0.7689

TABLE II
RESULTS WITH FOUR DISTRIBUTED DETECTION SITES

Distributed Detection Site	Detection Rate using Local Model	Detection Rate using Global Model
Site I	0.0074	0.9347
Site II	0.2822	0.7923
Site III	0.9351	0.9760
Site IV	0.2462	0.2500

250436 network intrusions. Note that the test data set is not from the same distribution as the training data, and it includes some intrusion types not existing in the training data. This makes the task more realistic. For more details of the intrusion data set, please refer to [21].

The proposed system is tested with three, four, and six distributed detection sites. To simulate a real distributed detection environment, the training data set is split into three, four, and six parts in the respective simulation, so that each site only have a small part of training data, with which an accurate intrusion detector is impossible to achieve.

In order to show the efectiveness of our parametric sharing models (10), we first test the performance of the local detector (8) obtained from the Adaboost training on each distributed detection site. After the local parametric models (10) are shared among the distributed detection sites, the GMMs (12) of the intrusion types from all the distributed sites are combined with the local ensemble weights (11), which are different among the distributed detection sites. Then the new detectors with global information are tested on the same test data set again. The detection rates of each local detector with three, four, and six distributed detection sites are listed in Table I, II, and III respectively. We can see great promotion on the detection rate of each distributed detection site after the local parametric models (10) are shared, with an average from 32.48% to 84.10%, although the sharing models have only a few parameters. The detection rates for each intrusion type from some of the distributed sites are listed in Tabel IV, which also shows the increase of detection ability of each local detection site after the sharing of local parametric models.

TABLE III
RESULTS WITH SIX DISTRIBUTED DETECTION SITES

Distributed Detection Site	Detection Rate using Local Model	Detection Rate using Global Model
Site I	0.0312	0.9324
Site II	0.0126	0.9945
Site III	0.2496	0.9767
Site IV	0.2457	0.8722
Site V	0.9271	0.9106
Site VI	0.0047	0.7165

TABLE IV
DETECTION RATES FOR EACH INTRUSION TYPE

Intrusion Type	Detection Rate No. 1 of 3 Sites Local/Global	Detection Rate No. 1 of 4 Sites Local/Global	Detection Rate No. 2 of 4 Sites Local/Global
bufferoverflow	0/0.0455	0/0.5	0.0909/0.2273
loadmodule	0/0.5	0.5/1	0.5/0.5
perl	0/0	0/1	1/0
neptune	1/1	0.0002/0.7712	0.9934/0.9981
smurf	0/0.9556	0/1	0/0.9570
guesspasswd	0.0985/0.0637	0.0016/0.7410	0.3723/0.26975
pod	0/0.4943	0.1264/1	0.9310/0.9310
teardrop	0.25/0.6667	0/0.75	0.4167/1
portsweep	1/1	0.7684/0.9718	0.9944/1
ipsweep	0/0.2516	0.9804/1	0.2974/0.8725
land	1/1	0/0.7778	0.8889/1
ftpwrite	0.3333/0.3333	0/0	0.6667/0.3333
back	0.0483/0.4791	0/0.8534	0/0.5947
imap	0/1	0/0	0/1
satan	0.9265/0.9927	0.2915/0.9571	1/0.9994
phf	0.5/1	0/1	0/1
nmap	0.8214/1	0.0595/1	0.8691/1
multihop	0.0556/0.5	0.0556/0.8333	0.4444/0.4444
warezmaster	0.0013/0.1979	0.0087/0.6348	0.2154/0.2366
warezclient	0/0	0/0	0/ 0
spy	0/0	0/0	0/0
rootkit	0/0.1538	0/0.4614	0.3846/0.4615
snmpgetattack	0.0005/0.0504	0/ 0.9960	0.7052/0.9932
named	0.4118/0.4118	0.1765/0.7647	0.2941/0.3529
xlock	0/0.2222	0/0.2222	0/0.2222
xsnoop	0/1	0/1	0.25/1
sendmai	0.1176/0.7059	0/0.8824	0.8235/0.8235
saint	0.8356/0.875	0.8940/0.9905	0.85608/0.9402
apache2	0.75315/0.9811	0/0.7557	0.3161/0.9584
udpstorm	0.5/1	0.5/1	0.5/1
xterm	0/0.0769	0.0769/0.8462	0.2308/0.0769
mscan	0.8927/0.9411	0.0446/0.6572	0.6762/0.9487
processtable	0.7405/0.7523	0.0224/0.7457	0.4058/0.9868
ps	0/0.25	0.0625/0.875	0.25/0.5625
httptunnel	0.8544/0.8734	0.1266/0.9494	0.9430/0.9430
worm	0/1	0/1	0/1
mailbomb	0/0.043	0/1	0/0.999
sqlattack	0/1	0.5/1	0/1
snmpguess	0/0.5586	0.0008/0.8803	0.5382/0.8030

The detection rate of the global detector (13) on the test data set is 93.66%, with a false alarm rate of 1.70%. Compared with the algorithm in [6] which gives a detection rate of 95% and a false alarm rate of 0.35%, the detection performance of our system is still very exciting as a distributed detection method for intrusions, as the detection results in [6] are obtained through some data preprocessing procedure. Otey et al. [6] process the intrusion data into bursts in the data set, and they mark an intrusion as detected if at leat one instance in a burst is flagged as an outlier. Furthermore, the high communication cost and the need of sharing the original network data make their algorithm inappropriate for the intrusion detection application. However, in our proposed system, no private data are shared among the distributed detection sites, and the only communication is the concise local parametric models (10), which have shown great effectiveness in Table I∼IV.

V. Conclusion

In this paper, we have proposed a framework for distributed detection of network intrusions based on a parametric model. The advantages of our framework are as follows. 1) The hybrid of Adaboost ensemble and GMM-based weak classifiers successfully overcomes the difficulties from the mixed-attribute property of network data. 2) The local parametric model is not only concise for communication among the distributed detection sites, but also can explicitly describe the distribution of the mixed-attribute data of each intrusion type, which is very suitable for information sharing. 3) The intuitively constructed global detector achieves a considerable improvement on the detection performance of each distributed detection site. 4) No original network data are shared in the framework, so that the data privacy is well protected.

Our future work will focus on the further improvement of detection accuracy of our system, and the investigating of incremental learning mechanism to fit for the dynamic streaming data environments.

Acknowledgment

This work is supported in part by the National Natural Science Foundation under Grants 60520120099 and 60672040 and in part by the National 863 High-Tech R&D Program of China under Grant 2006AA01Z453.

References

[1] D. Denning, "An intrusion-detection model," *IEEE Trans. on Software Engineering*, vol. 13, no. 2, pp. 222-232, February 1987.

[2] J. Li and C. Manikopoulos, "Novel statistical network model: The hyperbolic distribution," *IEE Proc. on Communications*, vol. 151, no. 6, pp. 539-548, December 2004.

[3] T. Peng, C. Leckie, and K. Ramamohanarao, "Information sharing for distributed intrusion detection systems," *Journal of Network and Computer Applications*, vol. 30, issue 3, pp. 877-899, August 2007.

[4] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," In *Proc. of IEEE Symposium on Security and Privacy*, pp. 120-132, May 1999.

[5] M. Qin and K. Hwang, "Frequent episode rules for Internet anomaly detection," In *Proc. of IEEE Int. Symposium on Network Computing and Applications*, pp. 161-168, 2004.

[6] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data mining and knowledge discovery*, vol. 12, no. 2-3, pp. 203-228, May 2006.

[7] Jr. J. M. Bonifacio, A. M. Cansian, A. C. P. L. F. De Carvalho, and E. S. Moreira, "Neural networks applied in intrusion detection systems," In *Proc. of IEEE Int. Joint Conf. on Neural Networks*, vol. 1, pp. 205-210, 1998.

[8] C. Zhang, J. Jiang, and M. Kamel, "Intrusion detection using hierarchical neural networks," *Pattern Recognition Letters*, vol. 26, no. 6, pp. 779-791, 2005.

[9] S. J. Han and S. B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Trans. on Systems, Man, and Cybernetics-Part B*, vol. 36, no. 3, pp. 559-570, June 2006.

[10] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," In *Proc. of Int. Joint Conf. on Neural Networks*, vol. 2, pp. 1702-1707, 2002.

[11] J. Mill and A. Inoue, "Support vector classifiers and network intrusion detection," In *Proc. of Int. Conf. on Fuzzy Systems*, vol. 1, pp. 407-410, 2004.

[12] J. Xian, F. Lang, and X. Tang, "A novel intrusion detection method based on clonal selection clustering algorithm," In *Proc. of Int. Conf. on Machine Learning and Cybernetics*, vol. 6, pp. 3905-3910, 2005.

[13] S. Jiang, X. Song, H. Wang, J. Han, and Q. Li, "A clustering-based method for unsupervised intrusion detections," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 802-810, May 2006.

[14] A. J. Hoglund, K. Hatonen, and A. S. Sorvari, "A computer host-based user anomaly detection system using the self-organizing map," In *Proc. of Int. Joint Conf. on Neural Networks*, vol. 5, pp. 411-416, 2000.

[15] H. G. Kayacik, A. Zincir-Heywood, and M. Heywood, "On the capability of an SOM based intrusion detection system," In *Proc. of Int. Joint Conf. on Neural Networks*, pp. 1808-1813, 2003.

[16] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hierarchical kohonenen net for anomaly detection in network security," *IEEE Trans. on Systems, Man, and Cybernetics-Part B*, vol. 35, no. 2, pp. 302-312, April 2005.

[17] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," In *Proc. of ACM Symposium on Applied Computing*, 2004.

[18] D. Song, M. I. Heywood, and A. N. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 3, pp. 225-239, June 2005.

[19] S. Parthasarathy, A. Ghoting, and M. E. Otey, "A survey of distributed mining of data streams," *Data streams: models and algorithms*, C. C. Aggarwal, Eds. New York: Springer-Verlag, November 2006.

[20] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, August 1997.

[21] S. Stolfo et al, The third international knowledege discovery and data mining tools competition [online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[22] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. on Information and systems security*, vol. 3, no. 4, pp. 227-261, November 2000.

[23] W. Hu and W. M. Hu, "Adaboost-based algorithm for network intrusion detection," *IEEE Trans. on Systems, Man and Cybernetics-Part B*, vol. 38, no. 2, pp. 577-583, April 2008.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 39, no. 1, pp. 1-38, 1977.

[25] W. Hu and W. M. Hu, "HIGCALS: a hierarchical graph-theretic clustering active learning system," In *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp. 3895-3900, October 2006.

[26] N. Oza, *Online Ensemble Learning*, PhD thesis, University of California, Berkeley, 2001.

[27] R. M. Neal and G. E. Hinton, "A new view of the EM algorithm that justifies incremental and other variants," *Learning in Graphical Models*, M. I. Jordan Eds. pp. 355–368, Kluwer Academic Publishers, 1998.

[28] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 246-252, June 1999.

[29] Dar-Shyang Lee, "Effective Gaussian mixture learning for video background subtraction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827-832, May 2005.