Octopus-IIDS: An Anomaly Based Intelligent Intrusion Detection System

Paulo M. Mafra, Vinicius Moll, Joni da Silva Fraga
Automation and Systems Departament (DAS)
Federal University of Santa Catarina (UFSC)
Florianopolis – SC – Brazil
Email: mafra,vmoll,fraga@das.ufsc.br

Altair Olivo Santin
Pontifical Catholic University of Parana (PUC-PR)
Curitiba – PR – Brazil
Email: santin@ppgia.pucpr.br

***Abstract*—The intrusion detection systems (IDS) are designed to identify unwanted attempts at manipulating, accessing or disabling of computer systems, mainly through a network, such as the Internet. Additionally, the IDSs can perform other functions like intrusion prevention (IPS), including proactive functions. A recurrent problem in intrusion detection systems is the difficulty to distinguish legitimate access from attacks. A lot of conventional IDSs are signature based, although they do not identify variations of these attacks nor new attacks.**

This paper presents an intrusion detection system model based on the behavior of network traffic through the analysis and classification of messages. Two artificial intelligence techniques named Kohonen neural network (KNN) and support vector machine (SVM) are applied to detect anomalies. These techniques are used in sequence to improve the system accuracy, identifying known attacks and new attacks, in real time. The paper also makes an analysis of the features used to classify data in order to define which of them are really relevant for each class of attack defined in our experiments.

***Keywords*-Internet Security; Intrusion Detection System; Artifitial Neural Network; Support Vector Machine;**

I. Introduction

The intrusion detection systems (IDSs) are usually classified into two main categories: signature based and anomaly based [1]. The signature based systems are designed to identify attacks that follow patterns previously recognized and reported by security experts, where each signature identifies a specific attack. In anomaly based IDSs, the normal behavior of the system or network traffic are represented and, for any behavior that varies over a pre-defined threshold, an anomaly activity is identified.

A weakness of signature based intrusion detection systems is the incapability of identifying new types of attacks or variations of known attacks. By the other side, in anomaly based IDSs, the number of false positives generated is higher than on those based on signatures. An important issue in anomaly based IDSs is how these systems should be trained, i.e., how to define what is a normal behavior of a system or network environment (which features are relevant) and how to represent this behavior computationally.

The training process requires a large amount of data and many artificial intelligence techniques can be employed, such as ANNs (Artificial Neural Networks). Artificial intelligence techniques have been used for both signature based and anomaly based intrusion detection. Among these techniques, we can cite the use of expert systems [2]. These systems employ a set of rules that represent patterns of known attacks or vulnerabilities to detect intrusions. Some data mining techniques have been used to identify normal patterns of behavior [3], [4].

Artificial neural networks had already been applied in IDSs [5], [6]. Most of these neural networks are composed of a set of input, some intermediate layers and one output. These networks have the capacity to identify patterns and variations of these patterns (variations of the same attack).

The main contribution of this paper is an intelligent intrusion detection system (IIDS) which uses two artificial intelligence techniques in sequence to better identify anomalies and to reduce the false positive rate present in related works. Another contribution is the analysis of the importance of each feature for each class of attack (DoS, Probe, R2L and U2R) present in the KDD Cup 1999 Data (The Third International Knowledge Discovery and Data Mining Tools Competition) in order to define which of this features are relevant to each class of attack. We also applied real network traffic to better evaluate the proposed system.

This paper presents a model for intrusion detection that can be applied into a real network traffic. This model makes use of two artificial intelligence techniques to reach low false positive rate. The paper is organized as follows: in Section II related works are presented. Section III describes the IDS developed model. In sequence, Section IV details the tests performed with the developed model and some results. Section V concludes the paper with some considerations about the proposed model as well as future work.

II. Related Works

The use of artificial neural networks (ANNs) in intrusion detection systems appears in several papers, such as [7], [8] and [9]. These works use SOM (*Self Organizing Maps*) and some variations to store data from the neural network training. The main idea of the artificial neural network approach for intrusion detection is the provision of an unsupervised classification method, that is fast and efficient for a large amount of data with many variables (source IP, destination IP, source port, destination port, size of packets, protocol, etc). One problem present in the artificial neural network approach is the time for training these networks, which is usually performed off-line. However, once trained, the time for analysis is considerably low.

Works involving neural networks to detect intrusions show promising results, such as decrease in the false positive rates and improve in the detection rate compared to other anomaly based IDSs. However, IDSs that use neural networks face the difficulty of training with real traffic and real attacks. Samples of real traffic, may have some kind of malicious traffic (noise) not identified. The application of malicious traffic in the neural network training period (for normal traffic) can affect the value of weights of the neurons, causing errors in the process of detection (depending on the learning rate). It is very difficult to classify a large amount of real traffic, identifying all existing attacks like malformed packets, fragmented packets, etc.

An intrusion detection system called PAYL [10] was developed with the objective of performing rapid detection, preferably on a gateway or in a front-end, avoiding the spread into other machines on the network. The method is based on anomalies and covers all network services. This method generates and analyzes mathematical models of the payloads (often based on the frequency of bytes in the payload) that should be delivered to the applications. The PAYL system “learns” and generates a profile of the payloads expected for each service. The detector captures the incoming payload and compares it with the profile that was generated for that service during the training period.

The POSEIDON [11] is an anomaly based IDS that analyzes the payload and the header of the packets. From the header, this system uses only the destination address and port number to build a profile of each service. The system is composed of two layers: the SOM and the PAYL model. The SOM is used to classify the data. The PAYL model has only one layer of nodes (neurons) where each neuron n is a vector of weights w_n associated. The size of the vector of weights is equal to the size of the largest data entry for a particular network service.

In [12] was proposed a hybrid classifier with multiple layers to detect intrusions, where the attacks are classified into three categories: denial of service (DoS), scan (PROBE) and “others”. The category “others” is divided into two sub-categories: local users trying to gain super user privileges (U2R) and remote to local (R2L) attacks.

Another recent work that addresses the use of artificial intelligence to detect intrusion is described by [13]. In this work are used Support Vector Machines (SVMs) to map the frequency of system calls made by each process on the machine. SVM is a powerful technique for solving problems related to learning, classification and prediction [14]. The developed system compares the system calls frequency with the mapped frequencies in the SVMs searching for discrepancies. In a dynamic environment, it is almost impossible to create profiles of users for determining the normal behavior. It is better to observe the behavior of processes instead of users. In this work, tests were performed using data from the DARPA 1998, comparing SVMs to other artificial neural networks. The results with SVMs presented a higher detection rate and lower false positive rates.

In another recent work, [15] presents a study comparing the use of support vector machine (SVM) and other techniques of data mining and neural networks for intrusion detection. Following the performed tests, the use of SVM had a higher detection rate than the techniques of data mining and slightly better than with other neural networks techniques. All the systems presented in this section make use of only one type of neural network for classification.

III. Octopus-IIDS Model

The intelligent intrusion detection systems (IIDSs) present in the literature apply only one type of neural network without reaching a good accuracy in intrusion detection. Such systems face problems in training the neural networks due to its large variance on the behavior of network traffic. By analyzing the characteristics of Kohonen networks [16], we can emphasize the ability to classify data in a generic way. Moreover, Support Vector Machine (SVM) has a very good accuracy when trained to separate the incoming data into only two classes. Based on this features, we developed a multi-layer, called Octopus-IIDS, using these two types of neural networks (Kohonen and Support Vector Machine).

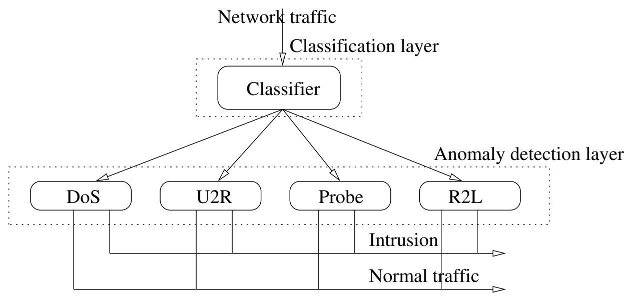

Figure 1. Architecture of Octopus

The main objective of octopus-IIDS is to provide an intelligent intrusion detection system (IIDS) that is accurate (with low false positive and false negative rates), flexible, tolerant to variations of attacks, adaptive to changes in the network, modular and that operates in real time. The octopus-IIDS generates models of the behavior of network traffic and perform its detection based on these models. Figure 1 shows the architecture of the proposed system.

This system is composed of two layers. The first, called **classifier** is responsible for collecting network traffic, analyze it and classify it into four categories: **DoS**, **U2R**, **Probe** or **R2L**, described in details in section III-B. In the second layer, the classified data will be applied to detectors specific to its classes (Anomaly Detection Layer). The second layer is responsible for saying if such data belongs to this specific class or do not. These layers are described in detail in the sections III-A and III-B respectively.

A. Classifier

The idea of the classifier is to make a pre-selection of the input traffic, through the analysis of characteristics related to the packets in a given period of time. This classifier is similar to some anomaly based IDSs presented in the related literature (one of these systems is presented by [12]). These classifiers have a high false positive rates and, in order to improve the detection rate, we decided to adjust the threshold of the detection algorithm to classify into one of the attack classes cited above at any evidence of attack, raising the false positive rate but reducing the false negative rate. Even the normal traffic will be classified in one of these attack classes. The output of the neural network classifier is send to another classifier, which is specialized in only one class of attack. This second classifier has the task of analyzing the input data and to identify more precisely what is an attack and what is not (considered as normal traffic). In the way, we can reduce the false positive rate of conventional IDSs and improve the detection rate.

The classifier is composed of a Kohonen neural network. The choice of this type of network was motivated by the characteristic of Kohonen neural networks in learning patterns automatically (unsupervised learning), the facility to separate known patterns (trained) and the generalization of patterns in the detection (it can detect variations of attacks).

We defined four categories (patterns) of traffic behavior to detect anomalies. The Kohonen neural network once trained for these four patterns (categories DoS, U2R, Probe and R2L) is able to separate suspect network traffic into these categories. We developed a Kohonen neural network with 41 inputs and 4 outputs. In order to calculate the distance between neurons, we adopted the Euclidean distance method $d = \sqrt{\sum_i (v_i - w_i)^2}$, where v_i is the vector of inputs and w_i is the vector of weights was applied for the learning process.

The use of Euclidean distance is ideal for random input data as the traffic to be analyzed in communication networks. During the learning process, the input data of the network must be applied several times (reinforcement) since the learning process needs to adjust the weights of the neurons.

For each entry of the classifier only one output neuron is activated. The value of each output neuron can vary from 0 to 1. If the output value is greater or equal to 0.8, the neuron will be activated. If the output value of the neuron is below than or equal to 0.2, it is not activated. If any neuron is able to be activated (with a value below 0.8), a function will force to the victory of any neuron which is between 0.2 and 0.8. The values usually used in Kohonen neural networks are 0.1 and 0.9. However, with such values the network tends to be more restrictive, identifying fewer cases of attacks but more precisely.

We chose to establish the values in 0.2 and 0.8 in the classifier to reduce errors in the classification of any possible attack. If the attack is directed to the right detector, this attack will probably be identified by the detector. Even if normal traffic is classified as some kind of attack, the detector will take the right decision. In tests, we also changed the values to 0.3 and 0.7 but the results were worse, classifying a large number of attacks in wrong classes.

B. Anomaly Detector

The anomaly detector in our model is composed of four Support Vector Machines (SVM), which receive traffic from the classifier. These SVMs deal with four categories:

- **DoS**: This class is responsible for identifying denial of service attacks, characterized by sending multiple requests to the same host and port in a short period of time.
- **U2R**: This attack class is characterized by the attempt of a local user become root.
- **Probe**: These attacks are characterized by the attempt to open connections to various ports of the same destination host in order to discover what services and versions are installed in the destination host.
- **R2L**: This category receives traffic flows which appear to be normal. These flows can be usually remote attacks to specific services.

For each category of attack mentioned above, a (SVM) specialized in the corresponding class of attack will have two options as output: normal traffic or malicious activity. We opted for the use of SVMs because the results obtained in [15] show that these networks are more efficient than other types of neural networks in the identification of anomalies. The SVM also bear a certain amount of noise in the input of the network (some attacks included in the data applied for training), without affecting his training. In the selection of configuration parameters, the SVM networks are less complex than other neural network models as the number of hidden layers, number of nodes for each layer and transfer functions. The wrong choice of some of these parameters may cause a degradation in performance of the network.

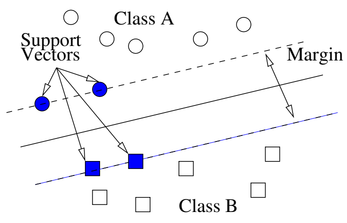

Figure 2. Separation of two classes by SVM

By using SVM networks, each category is represented by a hyperplane, as shown in Figure 2, defined by a number of support vectors, where the training data are separated into two classes: one for normal traffic and another for malicious activity (class A and B in the figure). These vectors form a sub set of training data used to define boundaries between the two classes (vectors filled in the picture). The limits may be expressed mathematically as: $w^T x + b = 0$, where w is the vector of weights, b is the bias and x is an input vector.

After the classification of data by the SVM, this data feeds the system (by adjusting the weights of the neurons of the classifier and SVM), allowing the system update. We use a low learning rate of 0.01 for this data, so that the system updates itself over the time but some mistakenly classified data will not affect the system accuracy.

We believe that the intrusion detection, using two layers in sequence, one for classification and another for decision-making is feasible, produces an increase in the detection rate and reduction in the false positive rates when compared with other anomaly based intrusion detection systems present in the literature. We also can use the Octopus-IIDS in the analysis of real traffic.

C. Prototype

A prototype has been developed, following the proposed model of octopus-IIDS for intrusion detection using artificial neural networks and SVMs. The prototype is focused on the classifier and on the anomaly detector, both were implemented using the Java programming language, in which the Kohonen neural network and the SVMs were implemented in the framework *Weka* (Waikato Environment for Knowledge Analysis) [1].

In this prototype, the input data (41 variables) is presented to the classifier and is retransmitted to the SVM network indicated by the classifier. The SVM network has two possible results: malicious activity or normal traffic.

In this prototype, we developed a format converter for adapting the input data in pcap format [2] to values that can be mapped into the input variables of the neural network (classifier). This prototype is available for download at http://www.das.ufsc.br/~mafra/octopus0.1.tar.gz.

D. KDD Data Analysis

In order to improve the proposed model, we made an evaluation of the important features for each class of attack, mapping from the 41 available features [17] which are relevant. Each class of attack has a distinct set of relevant features. In [18] were presented the degree of importance of the 41 features for each type of attack (smurf, snmpgetattack, warezmaster, etc). In this work, we identify which of the 41 features are relevant for each class of attack (DoS, Probe, R2L and U2R), since the attacks evolve over the time, but the classes of attacks remain the same.

[1]Machine Learning Software in Java. http://weka.wiki.sourceforge.net

[2]Standard format for capture of network packets. http://www.tcpdump.org/pcap/pcap.html

Based on this analysis, we can simplify the model, defining which features are really important for each class of attack. Thus, reducing the amount of memory and processing time required to run the system and improving the detection rate. Another positive factor is that we can use these features for analysis of real network traffic.

Table I
RELEVANT FEATURES FOR EACH CLASS OF ATTACK

Classes	Relevant Features
DoS	duration, protocol_type, service, src_bytes, is_guest_login, flag, count, srv_serror_rate, rerror_rate, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate
Probe	protocol_type, service, flag, src_bytes, count, srv_count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate
U2R	duration, protocol_type, service, flag, src_bytes, dst_bytes, logged_in, num_compromised, root_shell, num_root, num_file_creations, is_guest_login, count, srv_count, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, hot, dst_host_srv_count, dst_host_same_srv_rate, serror_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate
R2L	duration, protocol_type, service, flag, src_bytes, dst_bytes, logged_in, is_guest_login, count, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_srv_rerror_rate

The framework *Weka* was used to choose which features are relevant for each class of attack, where from all the features, the system pointed out those that have great influence in the SVM classification. Table I summarizes our results. In section IV, we evaluate the performance of the proposed model for intrusion detection, making comparative tests using the proposed model with all data and the proposed model with only those considered relevant. We also make a comparison of the achieved results with other approaches present in the literature.

IV. TESTS AND OUTCOMES

Tests with the prototype were performed on a machine with 4GB of RAM and two 1.6GHz Quad Core Intel Xeon processors. The KDD Cup 1999 Data traffic, available on the Internet [17], was used to test the presented model. According to our experiments, the minimum configuration required to run the tests should be a computer with a 1.6GHz processor and 1GB of memory.

The neural network classifier was trained with the four categories of attacks: DoS, User to Root (U2R), Probe, and Remote to Local (R2L) attacks. In the sequence, the four SVM anomaly detectors were trained with traffic specific to each class of attack. The vector of weights from the trained networks were stored in files for future use, without the need for further training. An important parameter to be chosen is the learning rate of the classifier. If the value is too low, it is needed many epochs and instances to train the network. If the value is too high, it does not converge. We performed tests with learning rates of 0.5 and 0.6.

After the training period of the system, tests were performed to measure the correct indication of the classifier to the anomaly detector (SVM). Tests were performed with 19.808 input data records randomly chosen from the KDD Cup 1999 (13.208 instances for training and 6.600 instances for testing). The learning rate used was 0.5 and 0.6 with 5.000 and 15.000 training epochs. Based on the results obtained, we can see that training the neural network (classifier) with learning rate of 0.6 is better than 0.5. The number of epochs for training (number of times the data records are presented to the neural network) is also important for the consolidation of a class of attack.

For the training of the Anomaly Detector, we performed some tests with the following algorithms: SMO Polykernel, SMO Normalized PolyKernel, SMO Pukkernel, LibSVM and SMO RBFKernel [3]. The SMO (Sequential Minimal Optimization) [19] is an efficient algorithm for SVM which repeatedly solves minimal size optimization. The minimal size optimization problem involves two Lagrange multipliers for finding the maximum /minimum of a function. The advantage of SMO lies in the fact that solving for two Lagrange multipliers can be done analytically. The difference among the tested algorithms is the kernel function used for classification. For instance, the SMO RBF Kernel is a radial base function. The tests with these algorithms were performed with the 41 features and with feature selection for each class of attack.

Table II
SVM RESULTS FOR EACH ANALYZED ALGORITHM WITHOUT FEATURE SELECTION

Training Function	DoS	Probe	U2R	R2L
SMO Polykernel	99,95 %	84,61 %	78,05 %	90,02 %
SMO Normalized Polykernel	99,85 %	83,46 %	78,05 %	85,45 %
SMO Pukkernel	99,75 %	91,29 %	75,61 %	94,35 %
LibSVM	90,52 %	81,30 %	53,66 %	98,02 %
SMO RBFKernel	98,19 %	74,94 %	53,66 %	80,63 %

Table II presents a comparison of the detection rate for each algorithm used for training the SVM analyzed by our prototype, using the 41 features.

In order to compare the effectiveness of the selected algorithms for each class of attack, tests were performed analyzing only the relevant features. Table III summarizes the results obtained. There is an improvement in the detection rate using only relevant features. This can be explained because with fewer entries and only the relevant ones, the

[3]The implementation of these algorithms are available at http://nlp.stanford.edu/nlp/javadoc/weka-3-2/weka.classifiers.SMO.html

Table III
SVM RESULTS FOR EACH ANALYZED ALGORITHM WITH FEATURE SELECTION

Training Function	DoS	Probe	U2R	R2L
SMO Polykernel	100 %	99,54 %	91,43 %	99,71 %
SMO Normalized Polykernel	100 %	99,42 %	88,57 %	99,13 %
SMO Pukkernel	100 %	99,59 %	82,86 %	99,61 %
LibSVM	99,59 %	88,10 %	62,86 %	94,10 %
SMO RBFKernel	100 %	98,72 %	62,86 %	97,05 %

SVM is less complex, without the other input entries that would not help in the detection and could also interfere in the classification result. The use of only relevant features have no effect on false positive or false negative rates. Based on this results, we chose the best training algorithm for each class of attack (SMOPolyKernel for DoS, U2R and R2L, SMOPukKernel for Probe).

Table IV
COMPARISON OF RESULTS BETWEEN IIDSS

IIDS	Average of detection	Max Deviation
Anomalous Payload-based IDS [11]	58,80 %	41,20 %
HPCANN [20]	77,49 %	22,53%
MADAM ID [21]	77,97 %	17,97%
Multi-level Hybrid Classifier [12]	89,19 %	22,52%
Octopus-IIDS	97,40 %	8,57 %

Table IV makes a comparison between the results obtained by Octopus-IIDS and some systems in the literature. All systems used the data from KDD CUP 99. We can see that the Octopus-IIDS obtained a good performance in all categories (low maximum deviation). The cited systems do not present some information as the number of data records used in the training period and details on the adjustments in the values of activation of the neurons in their networks.

Besides the KDD Cup 99 dataset is composed of not real data and has more than 10 years old, it is the most used in the related literature to evaluate IDSs prototypes. In order to better evaluate our system, we captured real network traffic from the Internet and applied to the Octopus-IIDS in order to verify its efficiency with real network traffic. During one hour, we collected 126.772 entries of normal traffic from an Internet server and 31.758 entries of attacks from a live honeypot configured to accept any kind of attack from the Internet. The normal traffic was distributed in the following ports: 25, 993 and 995 for email; 80 and 443 for web pages; and 53 for domain name system. We considered all the network traffic sent to the Internet server as normal traffic and all the traffic sent to the honeypot as malicious traffic.

In the training period, we randomly chose 84.937 entries of normal traffic and 21.278 entries of attacks. In the testing period, it were applied 41.835 entries of normal traffic and 10.480 entries of attacks (these entries are different from the training traffic). In this test we obtained a detection rate of 83,90 % with 9,72 % of maximum deviation. Since the related works did not use real network traffic we can not compare such results.

We believe that may have some malicious traffic into the normal traffic dataset, but in very small quantity since our network has others filters and IDSs to detect malicious traffic. Moreover our system were developed to support some traces of malicious traffic into the normal traffic dataset used for training the neural network without incluence the training process.

V. Concluding Remarks

This paper describes the development of an anomaly based intelligent intrusion detection model named Octopus-IIDS that makes use of artificial neural network and support vector machines. The use of these techniques are important to identify malicious activity through the analysis of network traffic, reducing the false positive rate and improving the detection rate.

The data from DARPA 98 and KDD Cup 99 are frequently used in tests of anomaly based IIDSs and can be used as parameters for comparison between these IIDSs, but the rate of attacks in these data is not natural (do not correspond to current reality of the Internet). About 80% of all instances correspond to attacks, usually with only one connection. The data for normal traffic are generated by simulators, without the presence of fragments of packets, disordered packets, etc. The Octopus-IIDS model can be applied in the analysis of real traffic because its structure allows the presence of some noisy traffic. Our test with real traffic showed the feasibility of the presented model, although there is some difficulties in comparing the results with other approaches. The overhead of the proposed system is increased by the use of two layers (about 60% more), when compared with a systems that use just one. In the developed model, the training process can be performed on an ongoing basis, with learning rate of 0.01, even in the presence of some misclassified traffic. Thus it is possible to keep the IIDS updated, even under the evolution of several applications that make use of the Internet.

References

[1] J. Allen, A. Christie, W. Fithen, J. McHugh, and J. Pickel, "State of the practice of intrusion detection technologies," in *CMU/SEI-99-TR-028*, 2000.

[2] T. Lunt, "Detecting intruders in computer systems," in *Conference on Auditing and Computer Technology*, 1993.

[3] S. M. Bridges and R. B. Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection," in *National Information Systems Security Conference*, October 2000.

[4] R. Mukkamala, J. Gagnon, and S. Jajodia, "Integrating data mining techniques with intrusion detection methods," in *Advances in Database and Information Systems Security*, 2000.

[5] G. Giacinto, F. Roli, and L. Didaci, *Fusion of multiple classifiers for intrusion detection in computer networks*, 2003.

[6] H. D. Lee, "Training a neural-network based intrusion detector to recognize novel attacks, systems, man and cybernetics," in *IEEE Transactions on Computer*, 2001, pp. 294–299.

[7] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proc. of the ACM symposium on Applied computing*, 2004, pp. 412–419.

[8] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an som based intrusion detection system," in *Proceedings of CNN*, 2003, pp. 1808–1813.

[9] J. Z. Lei and A. Ghorbani, "Network intrusion detection using an improved competitive learning neural network," in *Proceedings of CNSR*, 2004, pp. 190–197.

[10] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Proc. of RAID*, 2004, pp. 309–320.

[11] D. Bolzoni, S. Etalle, and P. Hartel, "Poseidon: a 2-tier anomaly-based network intrusion detection system," in *IEEE Workshop on Information Assurance*, 2006, pp. 220–237.

[12] C. Xiang and S. M. Lim, "Design of multiple-level hybrid classifier for intrusion detection system," in *Workshop on Machine Learning for Signal Processing*, 2005, pp. 117–122.

[13] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of svm and ann for intrusion detection," *Comput. Oper. Res.*, vol. 32, no. 10, pp. 2617–2634, 2005.

[14] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of IJCNN*, 2002, pp. 1702–1707.

[15] X. Haijun, P. Fang, W. Ling, and L. Hongwei, "Ad hoc-based feature selection and support vector machine classifier for intrusion detection," in *Proceedings of GSIS*, 2007.

[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Journal of the American Society for Information Science and Technology*, pp. 509–521, 1988.

[17] J. S. Stolfo, F. Wei, W. Lee, A. Prodromidis, and P. K. Chan, "knowledge discovery and data mining competition," 1999.

[18] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets," in *Proceedings of Privacy, Security and Trust*, 2005.

[19] J. Platt, B. Schlkopf, C. Burges, and A. Smola, *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1998.

[20] G. Liu, Z. Yi, and S. Yang, "A hierarchical intrusion detection model based on the pca neural networks," *Journal of Information Science and Technology*, pp. 1561–1568, 2006.

[21] W. Lee and S. Stolfo, "A framework for constructing features and models for idss," pp. 227–261, 2000.