ARTIFICIAL INTELLIGENCE APPROACHES FOR INTRUSION DETECTION.

Dima Novikov (Rochester Institute of Technology, Rochester, NY, dima.novikov@gmail.com), Roman V. Yampolskiy (University at Bufalo, Buffalo, NY, rvy@buffalo.edu, Leon Reznik (RIT, Rochester, NY, lr@cs.rit.edu)

ABSTRACT

Recent research indicates a lot of attempts to create an Intrusion Detection System that is capable of learning and recognizing attacks it faces for the first time. Benchmark datasets were created by the MIT Lincoln Lab and by the International Knowledge Discovery and Data Mining group (KDD). A number of competitions were held and many systems developed as a result. The overall preference was given to Expert Systems that were based on Decision Making Tree algorithms. This paper explores Neural Networks as means of Intrusion Detection. After multiple techniques and methodologies are investigated, we show that properly trained Neural Networks are capable of fast recognition and classification of different attacks at the level superior to previous approaches.

1. INTRODUCTION

One of the most promising areas of research in the area of Intrusion Detection deals with the applications of the Artificial Intelligence (AI) techniques. The most valuable feature of any AI system is the ability to learn automatically according to data inputs and outputs. This characteristic potentially can add more flexibility to Intrusion Detection Systems and remove the necessity to update the database of possible attacks constantly. At this point we talk not just about an IDS, but about Intelligent Intrusion Detection System (IIDS), which is capable of creating attack patterns, i.e. learning about new attacks, based on previous experience.

Multiple experiments were held by multiple research teams to apply AI techniques in Intrusion Detection. The main goal was to create a system that is capable of detecting different kinds of attacks. The researchers used DARPA and KDD Cups benchmark databases for training and detecting attacks in the experiments.

2. ARTIFICIAL INTELLIGENCE TECHNIQUES IN INTRUSION DETECTION

2.1 Neural Networks Approach

Neural Networks approach is one of the most interesting in this area. An increasing amount of research in the last few years has investigated the application of neural networks to intrusion detection. If properly designed and implemented, neural networks have the potential to address many of the problems encountered by rule-based approaches. Neural networks were specifically proposed to learn the typical characteristics of system's users and identify statistically significant variations from their established behavior. In order to apply this approach to Intrusion Detection, we would have to introduce data representing attacks and non-attacks to the Neural Network to adjust automatically coefficients of this Network during the training phase. In other words, it will be necessary to collect data representing normal and abnormal behavior and train the Neural Network on those data. After training is accomplished, a certain number of performance tests with real network traffic and attacks should be conducted.

Supervised Learning Model

Lippmann and Cunningham of MIT Lincoln Laboratory conducted a number of tests employing Neural Networks to misuse detection [10, 9]. The system was searching for attack-specific keywords in the network traffic. A Multi-layer Perceptron had been used for detection UNIX host attacks, and attacks to obtain root-privilege on a server. The system was trying to detect the presence of an attack by classifying the inputs into 2 (two) outputs (normal and attack). The system was able to detect 80% of attacks - 17 out of 20 attacks were identified. The main achievement of this system was its ability to detect old as well as new attacks not included in the training data.

Unsupervised Learning Model

There were a couple of systems, which used Self Organizing Maps for detecting intrusion. Luc Girardin's of UBILAB laboratory performed clustering of network traffic in order to detect attacks. A visual approach was

1-4244-0300-6/06/$20.00 ©2006 IEEE.

chosen for attack association [16]. SOM were employed to project network events on an appropriate 2D-space for visualization, then the network administrator analyzed them. Intrusions were extracted from the view by highlighting divergence from the norm with visual metaphors of network traffic. The main disadvantage of this approach is its need in interpretation of network traffic by an administrator or other authorized person to detect attacks.

Kayacik, Zincir-Heywood, and Heywood utilize KDD Cups data set for the experiments. They create three layer of employment [19]: First, individual SOM are associated with each basic TCP feature. This provides a concise summary of the interesting properties of each basic feature, as derived over a suitable temporal horizon. Second, integrates the views provided by the first level SOM into a single view of the problem. At this point, they use the training set labels associated with each pattern to label the respective best matching unit in the second layer. Third, the final layer is built for those neurons, which win for both attack and normal behaviors. This results in third layer SOMs being associated with specific neurons in the second layer. Moreover, the hierarchical nature of the architecture means that the first layer may be trained in parallel and the third layer SOMs are only trained over a small fraction of the data set.

	Normal	*DoS*	*Probe*	*U2R*	*R2L*
Level 2	92.4	96.5	72.8	22.9	11.3
Level 1	95.4	95.1	64.3	10.0	9.9

Table 1. Performance of 2 and 3 layer hierarchy on different categories

The table above describes the detection of attacks by category. In the table below we can see the individual attack detection rates.

Attack Name	*Level 2*	*Level 3*
Apache2	90.3	90.7
Httptunnel	58.9	20.9
Mailbomb	7.8	6.8
Mscan	90.2	60.9
Named	23.5	0.0
Processtable	59.4	47.6
Ps	0.0	0.0
Saint	79.1	78.7
Sendmail	5.9	11.8
Snmpgetattack	11.5	10.3
Udpstorm	0.0	0.0
Xlock	0.0	0.0
xsnoop	0.0	0.0
Xterm	23.1	30.8

Table 2. Detection rate of new attacks for 2-layer and 3-layer hierarchy

Hybrid Networks

Several researchers have combined MLP and SOM in their attempt to create an Intrusion Detection System. Cannady and Mahaffey of Georgia Technical Research Institute and Fox, Henning, and Reed have performed a research to apply Multi-Layer Perceptron model and Self-Organizing Map for misuse detection [26, 7, 14]. They have used a feed-forward network with back-propagation learning, which contained 4 fully connected layers, 9 input nodes and 2 output nodes (normal and attack). The network has been trained for a certain number of attacks. The network has succeeded in identifying attacks it was trained for.

Bivens believes that DOS and other network-based attacks leave a faint trace of their presence in the network traffic data. He has designed modular network-based intrusion detection system that analyzes TCP dump data to develop windowed traffic intensity trends, which detects network-based attacks by carefully analyzing this network traffic data and alerting administrators to abnormal traffic trends. It has been shown that network traffic can be efficiently modeled using artificial neural networks [2, 9, 10], therefore MLP was chosen to examine network traffic data. SOM has been used to group network traffic together to present it to the neural network, as SOM have

been shown to be effective in novelty detection [30, 16, 19].

The data that they have presented to the neural network consisted of attack-specific keyword counts in network traffic [22]. This system reminds a host-based detection system because it looks at the user actions. The Neural Network was created to analyze program behavior profiles instead of user behavior profiles [15]. This method identifies the normal system behavior of certain programs and compares it to the current system behavior. The author has used DARPA benchmark for the experiments. The prediction rate of the system is 24% - 100%. 100% has been achieved only with one attack in the training set – sshprocesstable [6, 4].

4.2 Rule-Based Approach

Agarwal and Joshi propose a two-stage general-to-specific framework for learning a rule-based model (PNrule) to learn classifier models on a data set that has widely different class distributions in the training data [1]. They utilized KDD Cups database for training and testing their system. The system was classifying the attacks into 4 main groups: Probing – information gathering, Denial of Service (DOS) – deny legitimate requests to the system, User-to-Root (U2R) – unauthorized access to local super-user or root, Remote-to-Local (R2L) – unauthorized local access from a remote machine.

As the result, the system performed very well on detecting Probing and DOS attacks identifying 73.2% and 96.6% respectively. 6.6% of U2R attacks were detected and 10.7% of R2L. False alarms were generated at a level of less than 10% for all attack categories except for U2R – an unacceptably high level of 89.5% false alarm rate was reported for this category

4.3 Decision-Tree Approach

Levin creates a set of locally optimal decision trees (decision forest) from which optimal subset of trees (sub-forest) is selected for predicting new cases [21]. 10% of KDD Cups database is used for training and testing. Data is randomly sampled from the entire training data set. Multi-class detection approach is used to detect different attack categories in the KDD data set. Just like Agarwal and Joshi [1], Levin tries to classify the data into four main categories: Probing, DOS, U2R, and R2L. The final trees give very high detection rates for all classes including the R2L in the entire training data set. In particular, 84.5% detection rate for Probing, 97.5% for DOS, 11.8% for U2R, and 7.32% for R2L. The following false alarm rates were detected for Probing, DOS, U2R and R2L attack categories respectively - 21.6%, 73.1%, 36.4%, and 1.7%.

4.4 Shared Nearest Neighbor and K-Means Approach

Ertoz used shared nearest neighbor technique (SNN) that is particularly suited for finding clusters in data of different sizes, density, and shapes, mainly when the data contains large amount of noise and outliers. All attack records are selected from the KDD training and testing data sets with a count of 10,000 records from each attack type: there are a total of 36 attack types from 4 attack categories. Also, 10,000 records were randomly picked from both the training and the testing data sets. In total, around 97,000 records were selected from the entire KDD data set. After removing duplicate KDD records, the data set size reduced to 45,000 records [13]. The author is utilizing two main clustering algorithms: K-Means, where the number of clusters is equal to 300, and SNN. K-Means performed very well on Probing, DOS, and R2L, detecting 91.8%, 98.75%, and 77.04% respectively. Detection rate for U2R is 5.6%. SNN performed in the following manner: 73.48% for Probing, 77.76% for DOS, 37.82% for U2R, and 68.15% for R2L. False alarms are not discussed by the author.

4.5 Parzen-Window Approach

Yeung and Chow propose a novelty detection approach using non-parametric density estimation based on Parzen-window estimators with Gaussian kernels to build an intrusion detection system using normal data only. This novelty detection approach was employed to detect attack categories in the KDD data set [29]. 30,000 randomly sampled normal records from the KDD training data set were used as training data set to estimate the density of the model. 30,000 randomly sampled normal records (also from the KDD training data set) formed the threshold determination set, which had no overlap with the training data set. The results were very high in most cases: 99.17% detection of Probing, 96.71% of DOS, 93.57% of U2R, and 31.17% of R2L. No false alarms information is available. The main advantage of this technique is its capability of detecting an attack, not just classifying the attacks.

4.6 Multi-Classifier Approach

Sabhnani and Serpen of the University of Toledo conduct a number of experiments with hybrid systems that

contained different approaches for attack classification. KDD Cups database is chosen for the experiments. The attacks are classified into four main groups, as was done by the researchers discussed in prior chapters: Probing – information gathering, Denial of Service (DOS) – deny legitimate requests to the system, User-to-Root (U2R) – unauthorized access to local super-user or root, Remote-to-Local (R2L) – unauthorized local access from a remote machine.

They highlight that most researchers employ a single algorithm to detect multiple attack categories with dismal performance in some cases. So, they propose to use a specific detection algorithm that is associated with an attack category for which it is the most promising [27]. Attributes in the KDD datasets had all forms – continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, preprocessing was required. The preprocessing includes the following steps:

Mapping symbolic-valued attributes to numeric-valued attributes. Symbolic features like protocol_type (3 different symbols), service (70 different symbols), and flag (11 different symbols) were mapped to integer values ranging from 0 to N-1 where N is the number of symbols. Attack names, such as buffer_overflow, were mapped to one of the five classes: Normal was mapped to 0, Probing was mapped to 1, DOS was mapped to 2, U2R was mapped to 3, 4 – R2L as described in [12],

Scaling: Each of the mapped features was linearly scaled to the range [0.0, 1.0]. Features having smaller integer value ranges like duration [0, 58329], wrong_fragment [0, 3], urgent [0, 14], hot[0, 101], num_failed_logins [0, 5], num_compromised [0, 9], su_attemptes [0, 2], num_root [0, 7468], num_file_creations [0, 100], num_shells [0, 5], num_access_files [0, 9], count [0, 511], srv_count [0, 511], dst_host_count [0, 255], and dst_host_srv_count [0, 255] were also scaled linearly to the range of [0, 1]. Logarithmic scaling (base 10) was applied to two features spanned over a very large integer range, namely src_bytes [0, 1.3 billion] and dts_bytes [0, 1.3 billion], to reduce the range to [0, 9.14]. All other features were either Boolean, like logged_in, having values (0 or 1), or continuous, like diff_srv_rate, in the range of [0, 1]. No scaling was necessary for these attributes

For the purpose of training different classifier models, all duplicate records are removed from the datasets. The total number of records in the original labeled training dataset is 972, 780 for normal; 41, 102 for Probe; 3,883,370 for DOS; 52 for U2R and 999 for R2L attack classes. All simulations are performed on a multi-user Sun SPARC machine, which has dual microprocessors, ULTRASPARC-II, running at 400 MHz. System clock frequency is equal to 100 MHz., the system had 512 MB of Ram and Solaris 8 operating system. 9 distinct pattern recognition and machine learning algorithms are tested:

MLP. 3 layers of Feed Forward Neural Network are implemented. Sigmoid is used as the transfer function and stochastic gradient decent with mean squared error function as the learning algorithm. The network has 41 (forty one) inputs; 5 outputs; 40 – 80 nodes in the hidden layer; 0.1 – 0.6 learning rate (0.1 the final rate); 500,000 samples in each epoch; and 30 – 150 epochs (60 the final number of epochs).

Gaussian classifier (GAU). This classifier assumes inputs are uncorrelated and distributions for different classes differ only in mean values. It is based on the Bayes decision theorem [11].

K-means clustering (K-M). This algorithm [11] positions K centers in the pattern space such that the total squared error distance between each training pattern and the nearest center is minimized.

Nearest cluster algorithm (NEA). It is a condensed version of K-nearest neighbor clustering algorithm [11]. Input to this algorithm is a set of cluster centers generated from the training data set using standard clustering algorithms like K-means, E & M binary split, and leader algorithm. In this case, the initial clusters were created using the K-means.

Incremental Radial Basis Function (IRBF). Can perform non-linear mapping between input and output vectors similar to RBF and MLP [17].

Leader algorithm (LEA). LEA partitions are a set of M records into K disjoint clusters (where $M \geq K$) [18]. First input record forms the leader of the first cluster. Each input record is sequentially compared with current leader clusters. If the distance measure between the current record and all leader records is greater than the threshold

(delta), a new cluster is formed with the current record being the cluster leader.

Hyper sphere algorithm (HYP). This algorithm creates decision boundaries using spheres in input feature space [3] [20]. Any pattern that falls within the sphere is classified in the same class as that of the center pattern. Spheres are created using an initial defined radius. Euclidian distance between a pattern and sphere centers is used to test whether a pattern falls in one of the current defined spheres.

Fuzzy ARTMAP (ART). Adaptive Resonance Theory mapping algorithm is used for supervised learning of multidimensional data [8]. It uses two ART's – ARTa and ARTb. ARTa maps features into clusters. ARTb maps output categories into clusters. There is a mapping from ARTa clusters to ARTb clusters that is performed during training.

C4.5 decision tree (C4.5). This algorithm was developed by Quinlan [28]. It generates decision trees using an information theoretic methodology. The goal is to construct a decision tree with minimum number of nodes that gives least number of misclassifications on training data. Divide and conquer strategy is utilized in this algorithm. The publicly available pattern classification software tool LNKnet is used to simulate pattern recognition and machine learning models [23]. The C4.5 algorithm is employed to generate decision trees using the software tool obtained at [5].

		Probe	DoS	U2R	R2L
MLP	*PD*	88.7%	97.2%	13.2%	5.6%
	FAR	0.4%	0.3%	0.1%	0.1%
GAU	*PD*	90.2%	82.4%	22.8%	0.1%
	FAR	11.3%	0.9%	0.5%	0.1%
K-M	*PD*	87.6%	97.3%	29.8%	6.4%
	FAR	2.6%	0.4%	0.4%	0.1%
NEA	*PD*	88.8%	97.1%	2.2%	3.4%
	FAR	0.5%	0.3%	0.1%	0.1%
RBF	*PD*	93.2%	73%	6.1%	5.9%
	FAR	18.8%	0.2%	0.4%	0.3%
LEA	*PD*	83.8%	97.2%	8.3%	1%
	FAR	0.3%	0.3%	0.3%	0.1%
HYP	*PD*	84.8%	97.2%	8.3%	1%
	FAR	0.4%	0.3%	0.1%	0.1%
ART	*PD*	77.2%	97%	6.1%	3.7%
	FAR	0.2%	0.3%	0.1%	0.1%
C4.5	*PD*	80.8%	97%	1.8%	4.6%
	FAR	0.7%	0.3%	0.1%	0.1%

Table 3. Multi – Classifier Results for separate algorithms

In the table above we can see the results of the experiments held by the authors. Here PD represents Probability of Detection, and FAR – False Alarm Rate. They state that the set of pattern recognition and machine learning algorithms tested on the KDD data sets offers an acceptable level of misuse detection performance for only two attack categories - Probing and DOS. On the other hand, all nine classification algorithms fail to demonstrate an acceptable level of detection performance for the remaining two attack categories that are U2R and R2L. Thus, [27] offers a multi-classifier model: they propose to have sub-classifiers trained using different algorithms for each attack category. They offer the best algorithm for each category: MLP for probing, K-M for DOS, K-M for U2R, GAU for R2L

The results are depicted in the table below, where we can see the improvement in the performance of the system overall.

		Probe	DoS	U2R	R2L
Multi-Classifier	*Pos Detection*	88.7%	97.3%	29.8%	9.6%
	False Alarms	0.4%	0.4%	0.4%	0.1%

Table 4. Multi-Classifier Results for the final system

3. OUR APPROACH

In the works we review in some cases accuracy of the classification is as low as 8.4%, which is totally not acceptable. The main problem with the approach they had chosen was that they used all attacks in the dataset, though many of those attacks did not have enough records for training, as we outlined after the data formatting and optimization took place. If an attack does not have enough presence (IMAP attack had only 12 records), it should not be used for training. Also, they grouped the attacks, what potentially can lead to a misdetection since not all of the attacks in the same group have identical signatures and patterns. Thus, a different approach was chosen to detect and classify attack. The main advantage of this approach was data formatting and the training dataset grouping, which allowed us to increase the accuracy rate up to 100% in some cases, and, the most important advantage, to achieve a high percentage of identification of the attacks that were not included into the training set [25].

The differences between our approach and the approach of other researchers are summarized below. First, we have chosen a different strategy in preprocessing. Before using the dataset, we made a thorough analysis of the given data. We found out that there are a lot of repeated records. It was obvious that some attacks, such as Smurf were taking more than 50% of the whole dataset, and some attacks have only 10 or even less records. To optimize the dataset, to make it appropriate for the training and testing, we wrote a tool that was capable of resolving mentioned above problems and prepare the dataset for the neural networks to use. So, the dataset was optimized: repeated records were removed, dataset was split into multiple files, one attack per file. After the statistics were computed, we chose those attacks that had more representation in the dataset, thus the attacks with insignificant number of records were omitted [25].

The second very important difference was the training set composition. After the records were converted into the neural network readable form, i.e. all values were mapped and scaled into the range [0:1], we created training sets, trying to keep the even distribution of the attacks in the set. In other words, if it is important to identify a normal behavior from the set of records, the records of the normal behavior has to take 50% of the training set. Other 50% should be evenly distributed in the group of attacks [24].

Third, we chose to classify each attack, when others were trying to classify the attacks into different groups. Group classification could potentially lead to a confusion, since though the attack belong to the same group, i.e. trying to achieve the same goal, they have different signatures, patterns, and set of actions, thus could be misclassified.

4. RESULTS AND CONCLUSIONS

First Stage - Known Attacks Detection

The first stage of the experiments consisted of 2 phases. First, only one attack was used in the training set. The distribution of an attack and normal records was 50% - 50%. Table 5 represents the results of these experiments. As it is shown, the accuracy of positive recognition is very high for both Neural Networks. All of the attacks have more than 90% of recognition. Most of them are very close to 100%, what is a very good expected result.

Attack Name	RBF Accuracy	RBF False Alarms	MLP Accuracy	MLP False Alarms
Smurf	100%	0	99.5%	0
Neptune	100%	0	100%	0
Satan	91%	7%	97.2%	2%
IP Sweep	99.5%	0	99.9%	0
Back	100%	0	100%	0

Table 5. One Attack Dataset Results

For the second phase of the first stage of the experiments, five different attacks were used in the training set. Normal behavior records, as it was mentioned before, was considered as an attack, thus total of six attacks were used in this stage. In order to proceed to the next level of the experiments, attack classification, it was important to prove that the attacks are distinguishable. Therefore, six different experiments were held to prove this idea. 50% of the training set consisted of the concentrated attack, i.e. the attack that had to be differentiated from the others. Other 50% were evenly distributed between other attacks, i.e. 10% per attack. For example, normal behavior records needed to be defined. 50% of the training set for this

assignment consisted of the records of normal behavior and other 50% contained records of Smurf, Neptune, Satan, IP Sweep, and Back attacks. All records were in random order. Table 6 demonstrates the results of this experiment. As shown in the table, the accuracy for differentiating the attacks is quite high for both Neural Networks. The lowest accuracy is 91% for Satan and the highest is 100% for Smurf, Neptune, and Back. These results let us make a conclusion that attacks can be differentiated, thus classified.

Attack Name	RBF Accuracy	RBF False Alarms	MLP Accuracy	MLP False Alarms
Smurf	100%	0	99.5%	0
Neptune	100%	0	100%	0
Satan	91%	7%	97.2%	2%
IP Sweep	99.5%	0	99.9%	0
Back	100%	0	100%	0
Normal	98.0%	1%	96.8%	2%

Table 6. Five Attack Dataset Results

Second Stage – Known Attacks Classification

For the second stage of the experiments Neural Networks with six outputs were used. At this level there was an attempt to create an Intrusion Detection System that is capable of classifying the attacks. A dataset of five attacks and normal behavior records were used. The attacks were evenly distributed in the dataset. Table 7 demonstrates the result of this experiment. As we can see the accuracy of classifying attacks is 93.2% using RBF Neural Network and 92.2% using MLP Neural Network. The results were very close and the difference is statistically insignificant. In most cases the Networks managed to classify an attack correctly. The false alarm rate (false positive) is very low in both cases, missed attacks rate (false negative) is not high either, and the misidentified attacks rate (misclassification of the attacks) is 5%-6%. Overall, it is possible to conclude that both Neural Networks managed to accomplish the second stage of the experiments and were capable of classifying the attacks. Therefore, the environment for the third stage of the experiments was set.

	Accuracy	False Alarms	Missed Attacks	Misidentified Attacks
RBF	93.2%	0.8%	0.6%	5.4%
MLP	92.2%	0	2.1%	5.7%

Table 7. Attacks Classification Results

Third Stage - Unknown Attacks Identification

For the third and the final stage of the experiments we used the trained Neural Network from the second stage. The Networks were trained to classify the following attacks: Smurf, Neptune, Satan, IP Sweep, Back, and Normal behavior records. At this point we proceeded with the most interesting and exciting phase of the experiments – untrained (unknown) attack classification and identification. As it was mentioned earlier, five attacks were chosen to be used for this purpose: Buffer Overflow, Guess Password, NMap, Teardrop, and Warezclient. Datasets of these attacks were sent into the trained Neural Networks. Table 8 demonstrates the results. As the table shows, RBF neural network managed to identify the unknown attacks as one of the trained attacks in most cases. As for the MLP Neural Network, it succeeded only with NMap and Guess Password attacks. In other cases it identified the attacks as normal behavior. Thus, RBF displayed more capabilities in identifying unknown attacks while MLP failed in some cases [24].

Attack Name	MLP	RBF
Buffer Overflow	53.3%	96.6%
Guess Password	96.2%	100%
NMap	99.5%	100%
Teardrop	1%	84.9%
Warezclient	8%	94.3%

Table 8. Unknown Attacks Classification Results

Overall, due to neural networks optimization, removing attacks with insignificant number of records, removing

repeated records, and taking a single attack approach instead of a group approach we managed to show better results than the results of other researchers.

Neural networks unlike expert systems allow for detection of previously unseen attacks. It makes them particularly attractive as an intrusion detection tool. Provided sufficient training data is available very high intrusion detection accuracy levels can be obtained as was demonstrated by our experiments. In the future we would like to investigate other types of neural networks which can be applied to the problem of intrusion detection and possible produce even better results.

5. REFERENCES

[1] R. Agarwal and M. Joshi, *PNrule: A New Framework for Learning Classifier Models in Data Mining*, Technical Report TR 00-015 (2000).

[2] A. Aussem, A. Mahul and R. Marie, *Queueing Network Modelling with Distributed Neural Networks for Service Quality Estimation in B-ISDN Networks*, Proceedings IEEE-INNS-ENNS International Joint Conference on Neural Networks (2000).

[3] B. G. Batchelor, *Pattern Recognition: Ideas in Practice*, Plenum Press (1978).

[4] A. Bivens, C. Palagiri, R. Smith, B. Szymanski and M. Embrechts, *Network-Based Intrusion Detection Using Neural Networks*, RPI (2001).

[5] C4.5, *http://www.rulequest.com*, (2004).

[6] J. Cannady, *Artificial Neural Networks for Misuse Detection*, Proceedings, National Information Systems Security Conference on Neural Networks (1998).

[7] J. Cannady and J. Mahaffey, *The application of artificial intelligence to misuse detection*, Proceedings of the 1st Recent Advances in Intrusion Detection (RAID) Conference (1997).

[8] G. A. Carpenter, S. Grossberg and N. Markuzon, *Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps*, IEEE Transactions on Neural Networks, vol. 3 (1992).

[9] R. Cunningham and R. Lippmann, *Detecting Computer Attackers: recognizing patterns of malicious stealthy behavior*, MIT Lincoln Laboratory - Presentation to CERIAS (2000).

[10] R. Cunningham and R. Lippmann, *Improving Intrusion Detection performance using Keyword selection and Neural Networks*, MIT Lincoln University (1998).

[11] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley (1973).

[12] C. Elkan, *Results of the KDD'99 Classifier Learning*, SIGKDD Explorations (2000).

[13] L. Ertoz, M. Steinbach and V. Kumar, *Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data*, Technical Report (2001).

[14] K. Fox, R. Henning and J. Reed, *A neural Network Approach Towards Intrusion Detection*, Proceedings of the 13th National Computer Security Conference (1990).

[15] A. Ghosh, A. Schwartzbard and M. Shatz, *Learning Program Behavior Profiles for Intrusion Detection*, Proceedings First USENIX Workshop on Intrusion Detection and Network Monitoring (1999).

[16] L. Girardin and D. Brodbeck, *A Visual Approach for Monitoring Logs*, 12th System Administration Conference (LISA '98) (1998), pp. 299-308.

[17] F. M. Ham, *Principles of Neurocomputing for Science and Engineering*, McGraw Hill (1991).

[18] J. A. Hartigan, *Clustering Algorithms*, John Wiley and Sons (1975).

[19] G. Kayacik, N. Zincir-Heywood and M. Heywood, *On the Capability of an SOM based Intrusion Detection System*, IEEE 0-7803-7898-9 (2003).

[20] Y. Lee, *Classifiers: Adaptive Modules in Pattern Recognition Systems*, Cambridge, MA: MIT (1989).

[21] I. Levin, *KDD-99 Classifier Learning Contest LLSoft's Results Overview*, SIGKDD Explorations, vol. 1 (2000).

[22] R. Lippmann and R. Cunningham, *Improving Intrusion Detection Performance Using Keyword Selection and Neural Network*, MIT Lincoln Laboratory (2001).

[23] LNKnet, *http://www.ll.mit.edu/IST/lnknet/index.html*, (2004).

[24] D. Novikov, R. V. Yampolskiy and L. Reznik, *Anomaly Detection Based Intrusion Detection*, *Third International Conference on Information Technology: New Generations (ITNG 2006)*, Las Vegas, Nevada, USA, April 10-12, 2006.

[25] D. Novikov., *Neural Networks to Intrusion Detection.*, *MS thesis*, Rochester Institute of Technology. Rochester, NY, October 2005.

[26] J. Planquart, *Application of Neural Networks to Intrusion Detection*, SANS Institute (2001).

[27] M. Sabhnani and G. Serpen, *Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context*, EECS, University of Toledo (2003).

[28] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD thesis (1974).

[29] D. Y. Yeung and C. Chow, *Parzen-window Network Intrusion Detectors*, Sixteenth International Conference on Pattern Recognition (2002).

[30] A. Ypma and R. Duin, *Novelty Detection using Self-Organizing Maps*, Progress in Connectionist-Based Information Systems, vol. 2 (1998).