International Journal of Computer Science and Applications, ©Technomathematics Research Foundation
Vol. 5, No. 2, pp 69 - 88

TRAFFIC ANALYSIS BASED IDENTIFICATION OF ATTACKS

DIMA NOVIKOV

Computer Science, Rochester Institute of Technology, 70-3521
Rochester, NY 14623, USA
dima.novikov@gmail.com

ROMAN V. YAMPOLSKIY[*]

Computer Science, University at Buffalo, CUBS
Buffalo, NY 14260 ,USA
rvy@buffalo.edu

LEON REZNIK

Computer Science, Rochester Institute of Technology, 70-3521
Rochester, NY 14623, USA
lr@cs.rit.edu

This paper is devoted to the problem of identification of network attacks via traffic analysis. Neural networks are chosen by us due to their capability to recognize an attack, to differentiate one attack from another, i.e. classify attacks, and, most important, to detect new attacks that were not included into the training set. Performed experiments demonstrate the advantage of our intrusion detection system compared to those created by the winner of the KDD Cup the leading data mining and knowledge discovery competition in the world. The results obtained indicate that it is possible to recognize attacks that the intrusion detection system never faced before on an acceptably high level.

Keywords: Attack; Intrusion Detection; Neural Network

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 Intrusion Detection System (IDS), but about an Intelligent Intrusion Detection System (IIDS), which is capable of creating attack patterns, i.e. learning about new attacks, based on previous experience (Novikov *et al.*, 2006).

Multiple experiments were perfomed by many 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 Defense Advanced Research Project

[*] Center for Unified Biometrics, 2145 Monroe Ave. #4, Rochester NY, 14618, USA, roman_y@hotmail.com .

Agency (DARPA) and Knowledge and Data Mining (KDD) Cups benchmark databases for training and detecting attacks in those experiments (Novikov *et al.*, 2006).

Most Intrusion Detection Systems (IDS) perform monitoring of a system by looking for specific "signatures" of behavior. However, using current methods, it is almost impossible to develop comprehensive-enough databases to warn of all attacks. This is for three main reasons. First, these signatures must be hand-coded. Attack signatures that are already known are coded into a database, against which the IDS checks current behavior. Such a system may be very rigid. Second, because there is a theoretically infinite number of methods and variations of attacks, an infinite size database would be required to detect all possible attacks. This, of course, is not feasible. Also, any attack that is not included in the database has the potential to cause great harm. Finally one other problem is that current methods are likely to raise many false alarms. So not only do novel attacks succeed, but legitimate use can actually be discouraged (Novikov *et al.* 2006b).

We investigate the benchmarks provided by the Defense Advanced Research Projects Agency (DARPA) and the International Knowledge Discovery and Data Mining Group (Darpa, 1998). These benchmarks and the experience of prior researchers are utilized to create an IDS that is capable of learning attack behavior and is able to identify new attacks without system update. In other words, we create a flexible system that does not need hand-coded database of signatures, and that can define new attacks based on pattern, not fixed rules provided by a third party. Neural networks are chosen as the means of achieving this goal. The use of neural networks allows us to identify an attack from the training set, also it allows us to identify new attacks, not included into the training set, and perform attack classification (Novikov *et al.*, 2006b).

2. Artificial Intelligence Techniques

2.1 Neural networks approach

An increasing amount of researchers have 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 establish 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.

2.1.1 Supervised learning model

Lippmann and Cunningham of MIT Lincoln Laboratory conducted a number of tests employing neural networks for misuse detection (Planquart, 2001; Rhodes *et al.* 2000). 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 two outputs: normal and attack. The system was able to detect 80% of attacks. The main achievement of this system was its ability to detect old as well as new attacks not included in the training data.

2.1.2 Unsupervised learning model

L. Girardin of UBILAB laboratory performed clustering of network traffic in order to detect attacks. A visual approach was chosen for attack association (Sabhnani and Serpen, 2003). Self Organizing Maps (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 *et al*. utilize KDD Cups data set for their experiments. They create three layer of employment (Kayacik *et al.*, 2003): First, individual SOM are associated with each basic Transmission Control Protocol (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.

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

	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

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

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

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%

2.1.3 Hybrid networks

Several researchers have combined Multi-Layer Perceptron (MLP) and Self-Organizing Map (SOM) in their attempt to create an intrusion detection system. Cannady *et al.* of Georgia Technical Research Institute and Fox *et al.* have investigated application of MLP model and SOM for misuse detection (Fox *et al.*, 1990; Cannady and Mahaffey, 1997; Planquart, 2001). 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 Denial Of Service (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 (Aussem *et al.*, 2000; Cunningham and Lippmann, 2000; Cunningham and Lippmann 2000b), 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 (Ypma and Duin, 1998; Girardin and Brodbeck, 1998; Kayacik *et al.*, 2003).

The data that they have presented to the neural network consisted of attack-specific keyword counts in network traffic (Cunningham and Lippmann, 2000). This system remains 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 (Ghosh et al., 1999). 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 (Cannady, 1998; Bivens *et al.*, 2002).

2.2 Rule-based approach

Agarwal *et al.* propose a two-stage general-to-specific framework for learning a rule-based model to learn classifier models on a data set that has widely different class distributions in the training data (Agarwal and Joshi, 2000). 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.

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.

2.3 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 (Levin, 2000). 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 (Agarwal and Joshi, 2000)

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%.

2.4 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 was reduced to 45,000 records (Ertoz *et al.*, 2001). 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.

2.5 Parzen-window approach

Yeung *et al.* propose a novel 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 novel detection approach was employed to detect attack categories in the KDD data set (Yeung and Chow, 2002). 30,000 randomly sampled normal records from the KDD training data set were used as training data 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 classifying the attack, not just detecting it.

2.6 Multi-Classifier Approach

Sabhnani *et al.* conducted a number of experiments with hybrid systems that contained different approaches for attack classification. KDD Cups database was chosen for the experiments. The attacks were classified into four main groups, as was done by the researchers discussed in prior sections: 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 (Sabhnani and Serpen, 2003). 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, R2L was mapped to 4 (Elkan, 2000).

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 were 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 were 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 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 (Duda and Hart, 1973).
- **K-means clustering (K-M).** This algorithm (Duda and Hart, 1973) 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 (Duda and Hart, 1973). 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 (Ham, 1991).
- **Leader algorithm (LEA).** LEA partitions a set of M records into K disjoint clusters (where M=K) (Hartigan, 1975). 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, 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 (Batchelor, 1978; Lee, 1989c). 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. Euclidean distance between a pattern and sphere centers is used to test whether a pattern falls in one of the currently defined spheres.
- **Fuzzy ARTMAP (ART).** Adaptive Resonance Theory (ART) mapping algorithm is used for supervised learning of multidimensional data (Carpenter, Grossberg *et al.* 1992). 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 (Werbos, 1974). 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 (LNKnet, 2004). The C4.5 algorithm is employed to generate decision trees using the software tool described in (C4.5, 2004).

Table 3. Multi-classifier results for separate algorithms.

		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%

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 namely: U2R and R2L. Thus, (Sabhnani and Serpen, 2003) 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.

Table 4. Multi-classifier results for the final system.

		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%

3. Experiments

In the works we review in some cases accuracy of the classification is as low as 8.4%, which is 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 (Novikov, 2005).

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 (Novikov, 2005).

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 even distribution of the attacks in the set. In other words, if it is important to identify normal behavior from the set of records, the records of the normal behavior have to comprise 50% of the training set. Other 50% should be evenly distributed in the group of attacks (Novikov *et al.*, 2006).

Third, we chose to classify each attack, when others were trying to classify the attacks into different groups. Group classification could potentially lead to 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.

3.1. Data

To conduct the experiments, it was decided to use the benchmarks of the International Knowledge Discovery and Data Mining group (KDD). These data are based on the benchmark of the Defense Advanced Research Projects Agency (DARPA) that was collected by the Lincoln Laboratory of Massachusetts Institute of Technology in 1998, and was the first initiative to provide designers of intrusion detection systems with a benchmark, on which to evaluate different methodologies (Darpa, 1998).

In order to collect these data, a simulation had been made of a factitious military network consisting of three target machines running various operating systems and services. Additional three machines were then used to spoof different IP addresses, thus generating traffic between different IP addresses. Finally, a sniffer was used to record all network traffic using the TCP dump format. The total simulation period was seven weeks.

Normal connections were created to profile those expected in a military network. Attacks fall into one of five categories: User to Root (U2R), Remote to Local (R2L), Denial of Service (DOS), Data, and Probe. Packets information in the TCP dump files were summarized into connections. Specifically, a connection was a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol. In 1999 the original TCP dump files were preprocessed for utilization in the IDS benchmark of the International Knowledge Discovery and Data Mining Tools Competitions (Hettich and Bay, 1999).

The data consists of a number of basic features: duration of the connection, protocol type, such as TCP, UDP or ICMP, service type, such as FTP, HTTP, Telnet, status flag, total bytes sent to destination host, total bytes sent to source host, whether source and destination addresses are the same or not, number of wrong fragments, number of urgent packets. Each record consists of 41 attributes and one target (Lee *et al.*, 1999; Lee *et al.*, 1999b). The target value indicates the attack name. In addition to the above nine basic features, each record is also described in terms of an additional 32 derived features, falling into three categories:

1. Content features: domain knowledge is used to assess the payload of the original TCP packets. This includes features such as the number of failed login attempts.

2. Time-based traffic features: these features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval.

3. Host-based traffic features: utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

In order to perform formatting and optimization of the data, a tool was written that is capable of completing such operations as computing data statistics, data conversion, data optimization, neural network input creation, and other data preprocessing related assignments. Based on the results produced by the Preparation Tool, we made the following classifications: each record consists of 41 fields and one target. The target value indicates the attack name. The data has 4,898,431 records in the dataset. 3,925,650 (80.14%) records represent attacks that fall into one of the five mentioned above categories. Total 22 attacks were identified. 972,781 (19.85%) records of normal behavior were found.

In order to perform formatting and optimization of the data, a tool was written that is capable of completing such operations as computing data statistics, data conversion, data optimization, neural network input creation, and other data preprocessing related assignments. Based on the results produced by the Preparation Tool, we made the following classifications: each record consists of 41 fields and one target. The target value indicates the attack name. The data has 4,898,431 records in the dataset. 3,925,650 (80.14%) records represent attacks that fall into one of the five mentioned above categories. Total 22 attacks were identified. 972,781 (19.85%) records of normal behavior were found.

Attributes in the KDD datasets contained multiple types: integers, floats, strings, booleans, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Therefore, preprocessing took place to transform the data into the most optimal format acceptable by the neural networks.

First of all, the dataset was split into multiple files and duplicate records were removed. Each file contained records corresponding to a certain attack or normal behavior. Thus, a library of attacks was created. It was done to achieve an efficient way to format, optimize, and compose custom training and testing datasets. Second, symbolic features like attack name (23 different symbols), protocol type (three 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. Third, a certain scaling had taken place: each of the mapped features was linearly scaled to the range [0.0, 1.0]. Features having integer value ranges like duration were also scaled linearly to the range of [0, 1]. 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.

Attacks with the most number of records were chosen to be in the training set. The following attacks were used to train and to test the neural networks: Smurf, Satan, Neptune, Ipsweep, Back. The following attacks were chosen for the unknown (not trained) set of attacks: Buffer_overflow, Guess_password, Nmap, Teardrop, Warezclient.

3.2 Neural networks based intrusion detection system experiments

It was decided to run the experiments in three stages. In stage one, it was important to repeat the experiments of other researchers and have the Neural Networks identify an attack. In stage two the experiment was aimed at a more complicated goal. It was decided to classify the attacks, thus, the Neural Networks had to determine not only the presence of an attack, but the attack itself. Stage three had to repeat the experiments of stage two, but in this stage a set of unknown attacks was added to the testing set. Stage three contains experiments of a higher complexity and interest.

We started by optimizing our NN in terms of utilized parameters and training set size and duration of training. For the MLP we have searched for: the optimal number of hidden nodes in terms of achieved accuracy, number of training epochs resulting in highest level of accuracy, and optimal Alpha and Beta values. For the RBF we have also investigated different width values of the radial basis centers. Figures 1, 2, and 3 show some results of our experiments on RBF neural network. Figure one show how the number of training epochs decreases the error rate of RBF. Figure two shows how the training set size influences the training time of the Radial Function based network. Finally, Figure 3 depicts the search for the optimal detection accuracy based on the variance of the training set size.

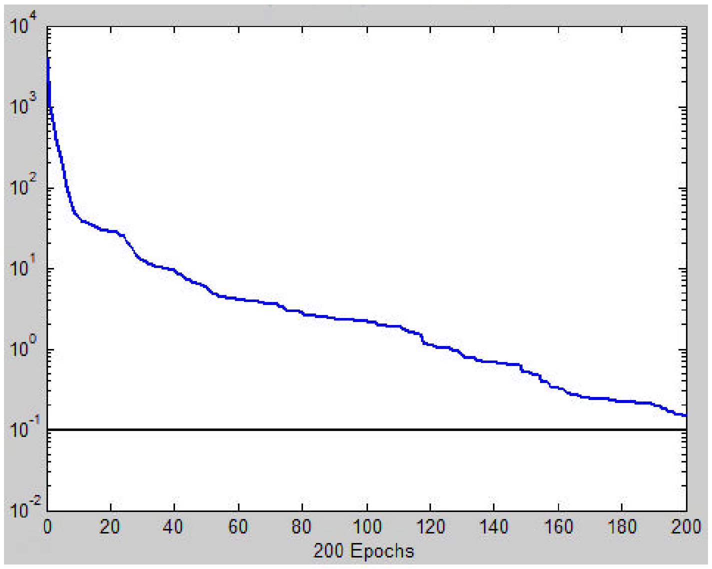

Fig. 1. Error rate decreases with the of training number epochs (RBF).

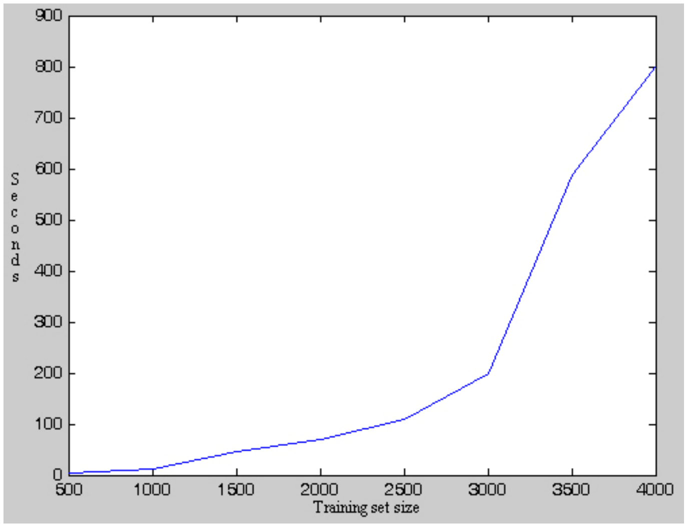

Fig. 2. Training time VS. the size of the training set for RBF.

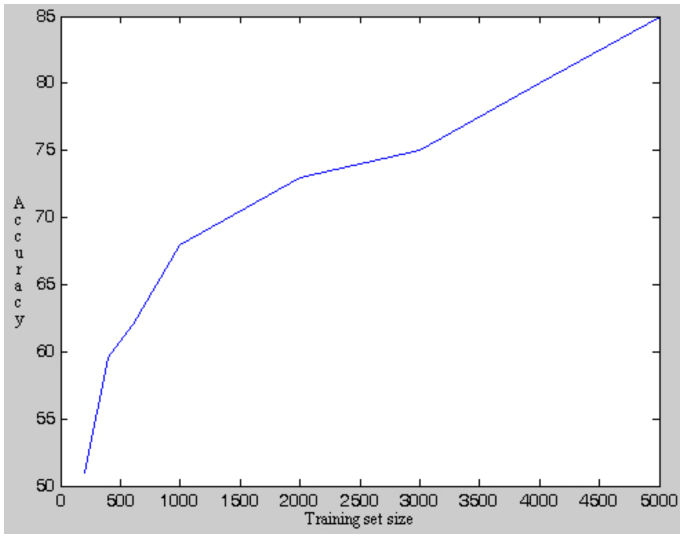

Fig. 3. Accuracy of RBF with respect to the training set size.

For the final experiments we have arrived at the following configurations: each Radial Bases Function (RBF) Neural Network had 41 inputs, corresponding to each attribute in the dataset, two outputs for attack detection (the first output for the presence of an attack – "yes", the second output for the normal behavior – "no"), or six outputs for attack classification (five outputs for the attacks, and the sixth output for the normal behavior), three layers (input, hidden, and output). The training set consisted of 4000 records. The attack and the normal behavior records were evenly distributed in the training set. The parameters of the Multiple Layer Perceptron (MLP) NN were: similar to those of RBF NN, but with some differences including: the hidden layer had 20 nodes, alpha = 0.7, beta = 0.8, “tansig” function is used in the input layer node, “purelin” in the hidden and output layer nodes. The training set consisted of 4000 records. The attack and the normal behavior records were evenly distributed in the training set.

3.3 Results

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% recognition. Most of them are very close to 100%, what is a very good and expected result (Yampolskiy, 2007).

Table 5. One attack dataset results.

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%

For the second phase of the first stage of the experiments, five different attacks were used in the training set. Normal behavior records 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 (Yampolskiy, 2007).

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.

Table 6. Five attack dataset results

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%

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 are capable of classifying the attacks.

Table 7. Attack Classification.

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

For the final stage of the experiments we used the trained NN 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 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: 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.

Table 8. Unknown attack detection.

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%

The winner of the last KDD intrusion detection competition Pfahringer, used C5 decision trees, the second-place performance was achieved by Levin using Kernel Miner tool, and the third-place contestants, Miheev *et al.* used a decision tree bas ed expert system (KDD, 1999). Also, we note the results of the most recent research made by Sabhnani *et al.* who used a multi classifier model to achieve even better results than the winner of the KDD Cup s contest (Sabhnani and Serpen, 2003).

Table 9 compares the mentioned above results. As we can see, in some cases accuracy of the classification is as low as 8.4%, which is 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 to achieve a high percentage of identification of the attacks that were not included into the training set.

Table 9.Results comparison.

		Probe	DoS	U2R	R2L
KDD Cup Winner	*Accuracy*	83.3%	97.1%	13.2%	8.4%
	False Alarms	0.6%	0.3%	0.1%	0.1%
KDD Cup Runner Up	*Accuracy*	83.3%	97.1%	13.2%	8.4%
	False Alarms	0.6%	0.3%	0.1%	0.1%
Multi-Classifier	*Accuracy*	88.7%	97.3%	29.8%	9.6%
	False Alarms	0.4%	0.4%	0.4%	0.1%

4. Conclusions

Many modern commercially used intrusion detection systems employ the techniques of expert systems that require constant updates from the vendors. This design makes the

IDS static, inflexible, and not capable of detecting new attacks without new patches. To improve the security, a lot of researchers put efforts to utilize artificial intelligence techniques in the area of intrusion detection, in order to create systems capable of detecting unknown attacks, and learning new attack signatures by themselves.

Benchmarks were created to standardize and compare the work of different investigators of this problem. Competitions were held to attract the attention of new researchers. In most cases decision-making trees were used. After extensive study, we decided to come up with a unique solution, and approached the problem with a new dataset formatting and optimization technique.

A library of attacks was created. This library was based on the benchmark provided by the MIT Lincoln Lab that was optimized by the KDD Cups. After the data was carefully formatted and optimized, it was decided to use and compare two different neural networks in attack detection and classification. Neural networks were chosen due to their abilities to learn and classify. Trained neural networks can make decisions quickly, making it possible to use them in real-time detection.

Both types of neural networks managed to perform well on the known set of attacks, i.e. attacks that they were trained to identify and classify. After new attacks were added to the testing set, i.e. attacks that were not included into the training set, Radial Basis Function Neural Network performed significantly better than Multiple Layer Perceptron with the detection rate between 80% and 100%, and the false alarm rate not greater than 2%.

When we compared these results to the results of previous work, it was notable that the chosen technique had its advantages. First of all, we managed to correctly detect the attacks. Second, classification of the trained attacks was successful with the rate of 90-100%. Third, and the most important, we were able to detect new unknown attacks, which were not included into the training set. The accuracy of detecting new unknown attacks was between 80% and 100%.

After performing our experiments we concluded that with appropriate data formatting, optimization, and dataset composition, neural networks display very good performance and potential in detecting and classifying trained attacks, as well as new unknown attacks that were not included into the training set.

Acknowledgments

This paper is based upon work supported by National Science Foundation Grant No. DGE 0333417 ”Integrative Geographic Information Science Traineeship Program”, awarded to the University at Buffalo.

References

-, -. (1998). DARPA, Intrusion Detection Evaluation. MIT Lincoln Laboratory, Available at: http://www.ll.mit.edu/ist/ideval.

-, -. (1999). "http://www-cse.ucsd.edu/users/elkan/clresults.html." KDD Cups 99 - Intrusion Detection Contest.

-, -. (2004). http://www.rulequest.com. C4.5.

-, -. (2004). LNKnet, Available at: http://www.ll.mit.edu/IST/lnknet/index.html.

Agarwal, R. and M. Joshi (2000). PNrule: A New Framework for Learning Classifier Models in Data Mining. Technical Report TR 00-015, Department of Computer Science, University of Minnesota.

Aussem, A., et al. (2000). 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, Como, Italy.

Batchelor, B. G. (1978). "Pattern Recognition: Ideas in Practice." Plenum Press.

Bivens, A., et al. (November 10-13, 2002). Network-Based Intrusion Detection Using Neural Networks. Artificial Neural Networks In Engineering, St. Louis, Missouri.

Cannady, J. (1998). Artificial Neural Networks for Misuse Detection. National Information Systems Security Conference on Neural Networks, Como, Italy.

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

Carpenter, G., et al. (1992). "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps." IEEE Transactions on Neural Networks vol. 3.

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

Cunningham, R. and R. Lippmann (2000 b). "Improving Intrusion Detection performance using Keyword selection and Neural Networks." Computer Networks 34(4): 597--603.

Duda, R. O. and P. E. Hart (1973). "Pattern Classification and Scene Analysis." Wiley.

Elkan, C. (2000). "Results of the KDD'99 Classifier Learning." ACM SIGKDD Explorations Newsletter 1(2).

Ertoz, L., et al. (2001). Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. Technical Report, University of Toledo.

Fox, K., et al. (1990). A Neural Network Approach Towards Intrusion Detection. Proceedings of the 13th National Computer Security Conference, Washington, D.C.

Ghosh, A., et al. (1999). Learning Program Behavior Profiles for Intrusion Detection. in Proceedings 1 st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California.

Girardin, L. and D. Brodbeck (1998). A Visual Approach for Monitoring Logs. 12th System Administration Conference (LISA ’98), Berkeley, CA.

Ham, F. M. (1991). "Principles of Neurocomputing for Science and Engineering." McGraw Hill.

Hartigan, J. A. (1975). "Clustering Algorithms." John Wiley and Sons.

Hettich, S. and S. D. Bay (1999). "The UCI KDD Archive." University of California, Department of Information and Computer Science.

Kayacik, G., et al. (2003). On the Capability of an SOM based Intrusion Detection System. Proceedings of the International Joint Conference on Neural Networks.

Lee, W., et al. (1999). Mining in a Data-Flow Environment: Experience in Network Intrusion Detection. In Proceedings of the 5th ACM SIGKDD, San Diego, CA.

Lee, W., et al. (1999 b). A Data Mining Framework for Building Intrusion Detection Models. IEEE Symposium on Security and Privacy, Okland, CA.

Lee, Y. (1989c). Classifiers: Adaptive Modules in Pattern Recognition Systems. MS Thesis, MIT, Department of Electrical Engineering and Computer Science.

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

Novikov, D., et al. (2006). Anomaly Detection Based Intrusion Detection. Third International Conference on Information Technology: New Generations (ITNG 2006), Las Vegas, Nevada, USA.

Novikov, D., et al. (2006b). Artificial Intelligence Approaches for Intrusion Detection. Long Island Systems Applications and Technology Conference (LISAT2006). , Long Island, New York.

Novikov., D. (2005). Neural Networks to Intrusion Detection. MS thesis, Rochester Institute of Technology. Rochester, NY.

Planquart, J. (2001). Application of Neural Networks to Intrusion Detection. SANS Institute, Available at: http://www.sans.org/reading_room/whitepapers/detection/336.php?portal=59ce6bc816da952ccdc3c878029b635a.

Rhodes, B., et al. (2000). Multiple Self-Organizing Maps for Intrusion Detection. National Security Systems Security Conference, Baltimore, MD.

Sabhnani, M. and G. Serpen (2003). Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003), Las Vegas, NV.

Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University.

Yampolskiy, R. V. (2007). Indirect Human Computer Interaction-Based Biometrics for Intrusion Detection Systems. The 41st Annual IEEE International Carnahan Conference on Security Technology (ICCST2007), Ottawa, Canada.

Yeung, D. Y. and C. Chow (2002). Parzen-window Network Intrusion Detectors. Sixteenth International Conference on Pattern Recognition, Quebec City, Canada.

Ypma, A. and R. Duin (1998). Novelty Detection using Self-Organizing Maps. Progress in Connectionist-Based Information Systems, Sp ringer.