The Application of Artificial Neural Networks to Misuse Detection: Initial Results

James Cannady

Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332
james.cannady@gtri.gatech.edu

James Mahaffey

Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332
jim.mahaffey@gtri.gatech.edu

Abstract

Misuse detection is the process of attempting to identify instances of network attacks by comparing current activity against the expected actions of an intruder. Most current approaches to misuse detection involve the use of rule-based expert systems to identify indications of known attacks. However, these techniques are less successful in identifying attacks which vary from expected patterns. Artificial neural networks provide the potential to identify and classify network activity based on limited, incomplete, and nonlinear data sources. We present the results of our ongoing research efforts into the application of neural network technology for misuse detection.

Keywords: Intrusion detection, misuse detection, neural networks, computer security.

1. Introduction

Because of the increasing dependence that companies and government agencies have on their computer networks the importance of protecting these systems from attack is critical. A single intrusion of a computer network can result in the loss or unauthorized utilization or modification of large amounts of data and cause users to question the reliability of all of the information on the network. There are numerous methods of responding to a network intrusion, but they all require the accurate and timely identification of the attack.

This paper presents an analysis of the applicability of neural networks in the identification of instances of misuse against a network. The results of tests conducted on a neural network-based proof-of-concept prototypes are also presented. Finally, the areas of future research that are being conducted in this area are discussed.

1.1 Intrusion Detection Systems

The timely and accurate detection of computer and network system intrusions has always been an elusive goal for system administrators and information security researchers. The individual creativity of attackers, the wide range of computer hardware and operating systems, and the ever-changing nature of the overall threat to targetted systems have contributed to the difficulty in effectively identifying intrusions. While the complexities of host computers already made intrusion detection a difficult endeavor, the increasing prevalence of distributed network-based systems and insecure networks such as the Internet has greatly increased the need for intrusion detection [14].

Most current approaches to the process of detecting intrusions utilize some form of rule-based analysis. Rule-based analysis relies on sets of predefined rules that are provided by an administrator, automatically created by the system, or both. Expert systems are the most common form of rule-based intrusion detection approaches [6, 18]. A number of non-expert system-based approaches to intrusion detection have been developed in the past several years [2, 3, 4, 7, 12, 15, 19, 20]. While many of these have shown substantial promise, expert systems remain the most commonly accepted approach to the detection of attacks.

1.2 Neural Networks

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. Neural networks can be generally divided into supervised and unsupervised training algorithms. In supervised training the network is provided data samples during training that include the desired output for each of the samples. In this way the network "learns" the desired output for a given input pattern. An example of this form of neural network architecture is the multi-level perceptron (MLP). The MLP is a highly interconnected series of neurons arranged in an input layer, an output layer, and one or more "hidden" layers. During training the neurons each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output in a layered feedforward topology. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds being the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Important issues in MLP design include specification of the number of hidden layers and the number of units in these layers. During training the MLP progresses iteratively, through a number of epochs. On each epoch, the training cases are each resubmitted back through the network, and target and actual outputs compared and the error calculated. This error, together with the error surface gradient, is used to adjust the weights. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs [7, 10].

Unsupervised training architectures learn from the training pattern without being provided with a desired output for each set. Self-Organizing Maps (SOM) are one of the most popular forms of unsupervised training neural network architectures, [13]. SOM's are competitive networks that provide a "topological" mapping from the input space to clusters. The SOM map is organized into a two-dimensional grid of neurons. A SOM tries to find clusters such that any two clusters that are close to each other in the grid have codebook vectors close to each other in the input space.

During training the SOM is presented with a series of data vectors. The SOM iteratively runs through a number of epoches. During each iteration the SOM selects the "winning" neuron based on the one which is nearest to the input vector and then adjust the winning neuron to be more like the input case. SOM networks are designed for classification tasks, not pattern recognition problems.

Unlike expert systems, which can provide the user with a definitive answer if the characteristics which are reviewed exactly match those which have been coded in the rulebase, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics which it has been trained to recognize. While the probability estimate determined by a neural network can be 100%, the accuracy of its decisions relies totally on the experience the system gains in analyzing examples of the stated problem.

1.3 Neural Network Intrusion Detection Systems

A limited amount of research has been conducted on the application of neural networks to detecting computer intrusions. Artificial neural networks offer the potential to resolve a number of the problems encountered by the other current approaches to intrusion detection. Artificial neural networks have been proposed as alternatives to the statistical analysis component of anomaly detection systems, [3, 4, 8, 17, 20]. Statistical Analysis involves the statistical comparison of current events to a predetermined set of baseline criteria. The technique is most often employed in the detection of deviations from typical behavior and determination of the similarly of events to those which are indicative of an attack [11]. Neural networks were specifically proposed to identify the typical characteristics of system users and identify statistically significant variations from the user's established behavior.

Artificial neural networks have also been proposed for use in the detection of computer viruses. In [7] and [7] neural networks were proposed as statistical analysis approaches in the detection of viruses and malicious software in computer networks. The neural network architecture that was selected for [7] was a SOM that was designed to learn the characteristics of normal system activity and identify statistical variations from the norm that may be an indication of a virus.

2. Initial Analysis of Approach

In an effort to determine the applicability of neural networks to the problem of misuse detection we conducted a series of experiments utilizing simulated network traffic in increasingly complex prototypes. The experiments were designed to determine if indications of attacks could be identified from typical network traffic, but they were not intended to completely resolve the many issues involved in effectively applying neural networks to misuse detection.

2.1 Neural Network IDS Prototypes

The prototype development processes consisted of designing the neural networks and preparing network data for use in the training and testing of the neural networks. The prototypes were developed using NeuralWorks Professional II/Plus™ from NeuralWare.

2.1.1 MLP Prototype

The first prototype neural network was designed to determine if a neural network was capable of identifying specific events that are indications of misuse. Neural networks had been shown to be capable of identifying TCP/IP network events in [21], but our prototype was designed to test the ability of a neural network to identify indications of misuse. The prototype utilized a MLP architecture that consisted of four fully connected layers with nine input nodes and two output nodes. The number of hidden layers, and the number of nodes in the hidden layers, was determined based on the process of trial and error. Each of the hidden nodes and the output node applied a Sigmoid transfer function (**1/(1 + exp (-x))**) to the various connection weights. The neural network was designed to provide an output value of 0.0 and 1.0 in the two output nodes when the analysis indicated no attack and 1.0 and 0.0 in the two output nodes in the event of an attack.

Data for training and testing the first prototype was generated using the RealSecur e™ network monitor from Internet Security Systems, Inc. The RealSecure™ monitor was configured to capture the data for each event that would be consistent with a network packet frame, (e.g., source address, destination address, packet data, etc.). The results of the RealSecure™ analysis on each event were also collected. These analyses classified the events as Low, Medium, or High, based on the threat to the target system posed by the event, (i.e., a high priority event is one which typically constitutes an alert). Approximately 10000 individual events were collected by RealSecure™ and stored in a Microsoft Access™ database, of which approximately 3000 were simulated attacks.

Three levels of preprocessing of the data were conducted to prepare the data for use in the training and testing of the neural network. In the first round of preprocessing nine of the event record data elements were selected from the available set, (e.g., protocol ID, source port, destination port, source address, destination address, ICMP type, ICMP code, raw data length, raw data). The nine elements were selected because they are typically present in network data packets and they provide a thorough description of the information transmitted by the packet.

The second part of the preprocessing phrase consisted of converting three of the nine data elements (ICMP type, ICMP code and raw data) into a standardized numeric representation, (the other six components were already in a numerical format that could be input into a MLP). The process involved the creation of relational tables for each of the data types and assigning sequential numbers to each unique type of element. This involved creating queries for each of the three data types and loading those results into tables that assigned a unique integer to each entry. These three tables were then joined to the table that contained the event records. A query was then used to select the remaining six of the nine original elements (ProtocolID, Source Port, Destination Port, Source Address, Destination Address, and Raw Data Length) and the unique identifiers that pertain to the three processed elements (ICMP Type ID, ICMP Code ID, and Raw Data ID). Two additional elements, (0.0, 1.0 for an attack and 1.0,0.0 for a non-attack event), were added to each record based on a determination of whether this event represented part of an attack on a network. These elements were used during training as the target output of the neural network for each record. The third round of data preprocessing involved the conversion of the results of the query into an ASCII comma delimited normalized format that could be used by the neural network.

The training of the MLP was conducted using a backpropagation algorithm for 10,000 iterations of the selected training data. Like the feed-forward architecture of the neural network, the use of a backpropagation algorithm for training was based on the proven record of this approach in the development of neural networks for a variety of applications [10]. Of the 9,462 records which were preprocessed for use in the prototype, 1000 were randomly selected for testing and the remaining were used to train the system. At the conclusion of the training the following results were obtained:

- `Training data root mean square error = 0.058298`
- `Test data root mean square error = 0.069929`
- `Training data correlation = 0.982333`
- `Test data correlation = 0.975569`

2.1.2 MLP Prototype Results

After the completion of the training and testing of the MLP neural network the various connection weights were frozen and the network was interrogated. Three sample patterns containing "normal" network events and a single simulated attack event (e.g., ISS scans, Satan scans, SYNFlood, etc.) were used to test the neural network. The MLP was able to correctly identify each of the imbedded attacks in the test data, (Figures 1-3).

While this prototype was not designed to be a complete intrusion detection system, the results clearly demonstrate the potential of a neural network to detect individual instances of possible misuse from a representative network data stream.

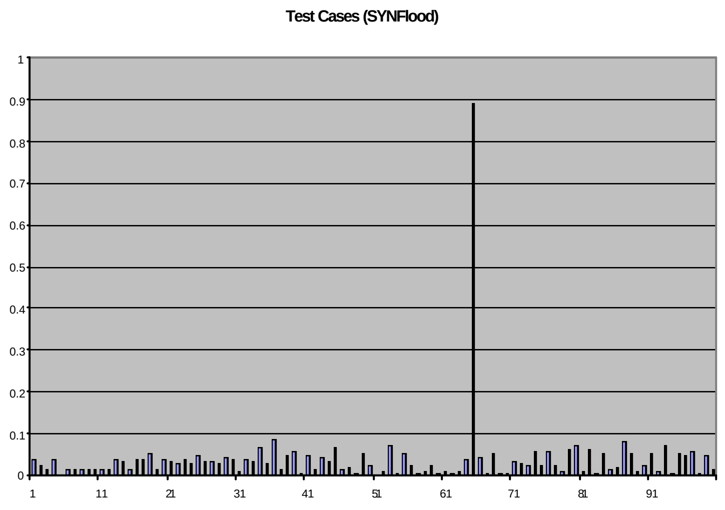

Figure 1 : SYNFlood Attack Test

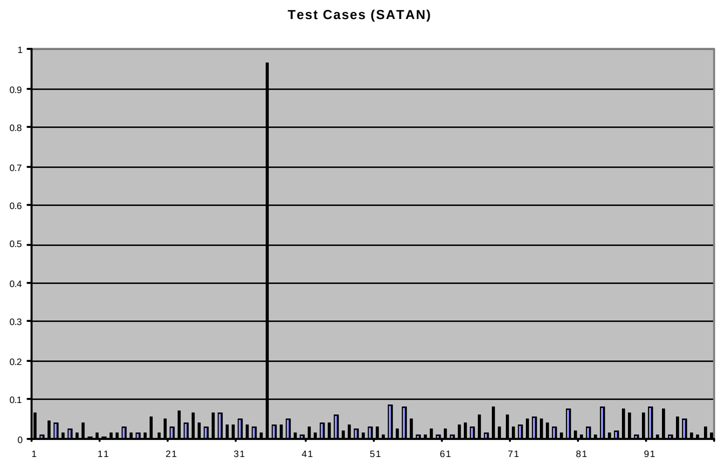

Figure 2 : SATAN Attack Test

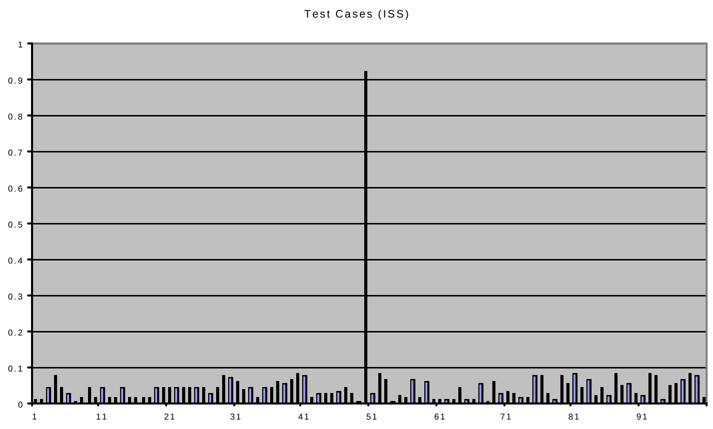

Figure 3 : ISS Scan Attack Test

2.1.3 Hybrid MLP/SOM Prototype

While the first prototype demonstrated the ability of a neural network to identify specific events that may be a part of an intrusion, most attacks are characterized by a series of events. Each of these events reviewed individually may appear normal, but when analyzed as a subtle probing, spread out over the time-development of the network traffic, an evolving pattern can be recognized imbedded in ordinary traffic.

The second prototype neural network approach to misuse detection was designed to address the need to identify these temporally dispersed and possibly collaborative attacks in a simulated data stream. Temporally dispersed attacks are those conducted by a single attacker over an extended period of time. Collaborative attacks are conducted by multiple attackers working in concert to achieve a single intrusion. Each of the attackers actions taken individually may appear innocuous with the attack becoming apparent only if all of the events are viewed together. Both of these types of attack may be very complex. As a result an enhanced pattern recognition approach was designed and tested to identify series of events that constitute an attack.

The problem was modeled by building two sets of network-traffic data. The first set contains randomly imbedded simulated attacks, and the second set is normal traffic. The data streams were divided into frames of 180 events each, or 3 minutes of simulated events if the traffic rate is one event per second, so that units of one data-frame could be evaluated for the presence of attack evidence. Each event was defined as the first 50 characters of a network contact, converted to ASCII equivalent normalized numbers, plus a normalized representation of the destination port. The first 50 characters of the raw data were used because most of the raw data field in the collected network data contained ≤ 50 characters. In addition, the use of 50 characters provided a consistent data set length for the prototype. A similar method of analyzing raw data was described in [9]. The destination port was utilized to clarify whether an even is telnet, ftp, etc. Each of the two network-traffic sets consisted of 50 frames.

Nine different attack patterns were formulated and imbedded in the first traffic set. An example of an attack pattern is a brute-force attack on an FTP server, beginning with the use of a username followed by three password attempts. Most servers will disconnect after three attempts, so the username is required again. Nine failed attempts to log in under the same user name within three minutes are clearly an indication of a problem. The nine events characterizing this type of attack would be:

- FTP Username
- FTP password attempt
- FTP password attempt
- FTP password attempt
- Same FTP Username
- FTP password attempt
- FTP password attempt
- FTP password attempt
- Same FTP Username
- FTP password attempt
- FTP password attempt
- FTP password attempt

No single event looks suspicious. Similarly, even two or three events may not appear unusual. However, taken together, in sequence, the collection of nine events is an indication of a problem. A further complication is that these events, though in sequence, can be interspersed among other, normal events. The series of events constituting an attack can be widely dispersed within the network stream.

2.1.4 Hybrid Neural Network Design

A feed-forward network with back-propagation learning was built to distinguish normal 3-minute (180-event) traffic frames from 3-minute traffic frames containing attacks. Two outputs were used; one for attacked frame, one for normal frame. So an output vector of 1.0,0.0 indicates an attack, 0.0,1.0 indicates no attack.

Each event in a 180-event frame consisted of a number-coded sequence of 50 characters plus a destination code, and these data had to be processed down into 180 numbers, in which each number coded a 50-character event into a type, or a into some category consistent with similar events. For this purpose, a pre-loader was built, using a self-organizing map (SOM). With some experimentation a 25x20 SOM was built to sort the 50-character events into a 500-node matrix, in which events of similar numeric character would be clustered. X and Y values on the matrix were then coded into one number per event using the formula:

$$z = ((\#columns * column\ value) + (\#rows + row\ value))/\#nodes$$

Each resulting node-number was normalized between -1.0 and 1.0. The validity of the SOM was tested by knowledge of where attack files occurred in the training files, so that it could be seen easily where on the matrix attacks clustered.

2.1.5 Hybrid MLP/SOM Results

The 180-input feed-forward network was trained using the event-data preprocessed by the SOM. The 50 cases containing one attack (a series of nine or more FTP login attempts with the same userID) per set was interleaved with the 50 cases of normal traffic containing no attacks, then this combined set was divided into two similar sets of 50 cases each. One case-set was used for active training, the other as a test set.

This configuration trained rapidly and without ambiguity to an insignificant root-mean-squared (RMS) error, indicating the possibility of finding seemingly subtle, time-evolving attack patterns interspersed randomly in ordinary Internet traffic.

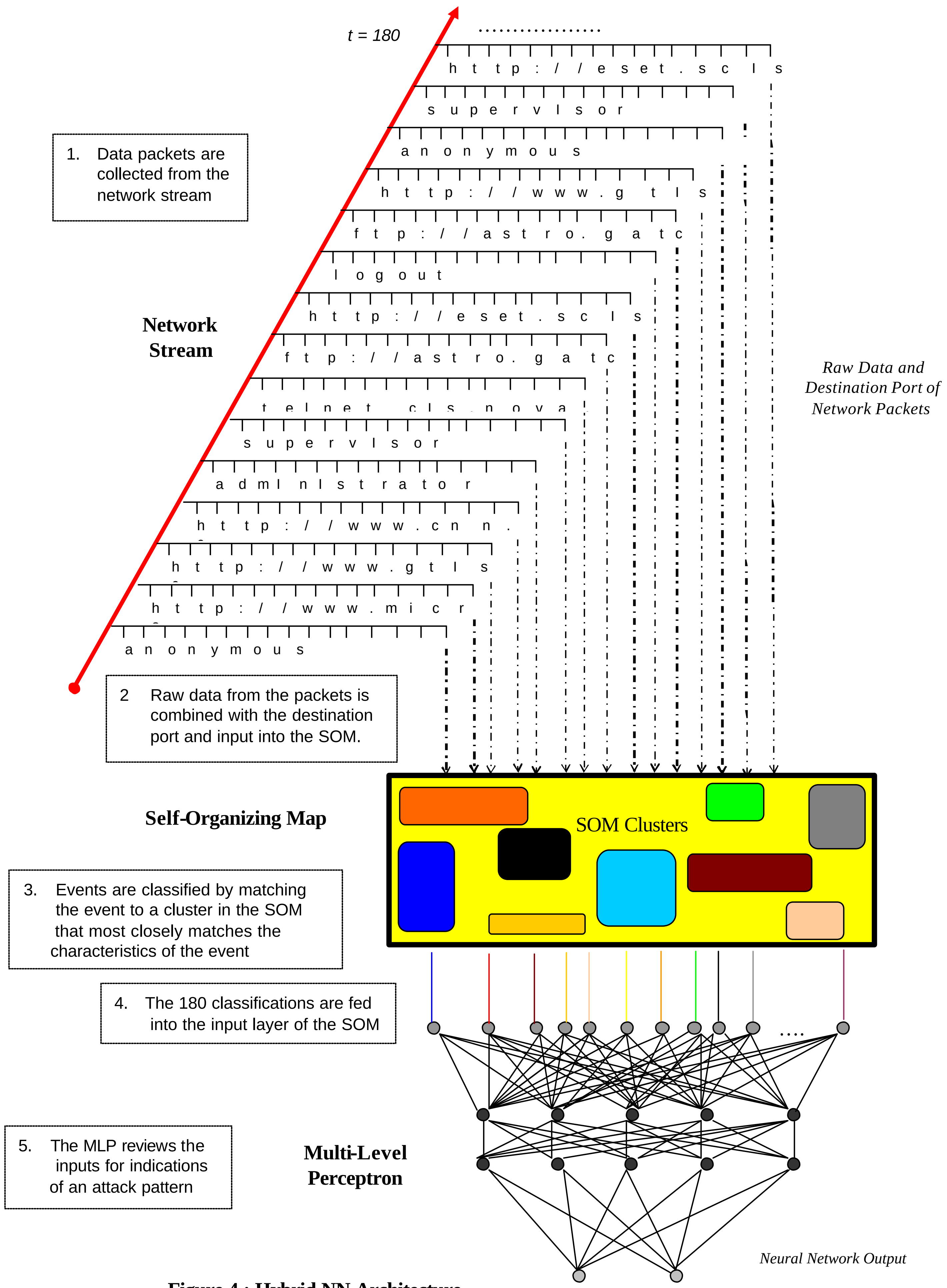
t = 180
h t t p : / / e s e t . s c l s
s u p e r v i s o r
a n o n y m o u s
h t t p : / / w w w . g t l s
f t p : / / a s t r o . g a t c
l o g o u t
h t t p : / / e s e t . s c l s
f t p : / / a s t r o . g a t c
t e l n e t c l s . n o v a
s u p e r v i s o r
a d m i n i s t r a t o r
h t t p : / / w w w . c n n .
h t t p : / / w w w . g t l s
h t t p : / / w w w . m i c r
a n o n y m o u s
1. Data packets are collected from the network stream
Network Stream
Raw Data and Destination Port of Network Packets
2 Raw data from the packets is combined with the destination port and input into the SOM.
Self-Organizing Map
SOM Clusters
3. Events are classified by matching the event to a cluster in the SOM that most closely matches the characteristics of the event
4. The 180 classifications are fed into the input layer of the SOM
5. The MLP reviews the inputs for indications of an attack pattern
Multi-Level Perceptron
Neural Network Output

Figure 4 : Hybrid NN Architecture

2.1.6 Hybrid SOM/MLP Tests

After completing the training and initial tests of the hybrid neural network architecture four tests were conducted to evaluate the ability of the design to identify patterns in a simulated data stream. Since the hybrid neural network was trained to recognize patterns of nine or more unsuccessful FTP login attempts the tests were conducted using data sets containing 6, 12, 18, and 0 FTP attempts respectively from a total of 180 event classifications per data set. The FTP events were widely dispersed throughout the data sets in an effort to thoroughly test the capability of the neural network to correctly identify FTP attacks.

The hybrid neural network was able to correctly classify each of the test cases. The prototype calculated the values for the test cases between .02/.98 when no FTP patterns were included in the data set to .972/.028 when eighteen FTP events were included in the data set, (Figure 5). These results provide a positive demonstration of the ability of the hybrid neural network architecture to detect complex instances of misuse.

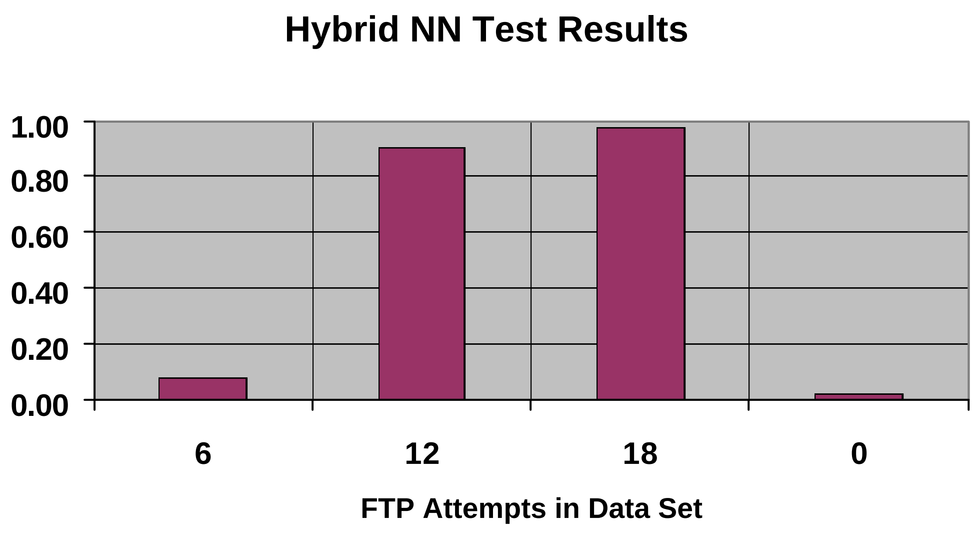

Figure 5 : Hybrid MLP/SOM Tests

3. Conclusions

Research and development of intrusion detection systems has been ongoing since the early 1980's and the challenges faced by designers increase as the targeted systems because more diverse and complex. Misuse detection is a particularly difficult problem because of the extensive number of vulnerabilities in computer systems and the creativity of the attackers. Neural networks provide a number of advantages in the detection of these attacks. The early results of our tests of these technologies show significant promise, and our future work will address the remaining issues in the development of a complete neural network-based intrusion detection system.

4. Further Work

The preliminary results from our proof-of-concept prototypes give a positive indication of the potential offered by this approach, but a significant amount of research remains before these technologies can demonstrate all of the required capabilities of an effective intrusion detection system. The most effective neural network architecture is an issue that must be addressed. The multi-level perceptron and the self-organizing map components of our prototypes offer a number of advantages in this type of problem, but there are several other neural network architectures that may better suited to detecting misuse.

In addition, an effective neural network-based approach to misuse detection must be highly adaptive. Most neural network architectures must be retrained if the system is to be capable of improving its analysis in response to changes in the input patterns, (e.g., “new” events are recognized with a consistent probability of being an attack until the network is retrained to improve the recognition of these events). Several approaches are being investigated for possible use in an intrusion detection system.

Finally, regardless of the initial implementation of a neural network-based intrusion detection system for misuse detection it will be essential for the approach to be thoroughly tested in order to gain acceptance as a viable alternative to expert systems. Work has been conducted on taxonomies for testing intrusion detection systems ([1, 16]) that offer a standardized method of validating new technologies. Because of the questions that are certain to arise from the application of neural networks to intrusion detection, the use of these standardized methods is especially important.

5. References

[1] Chung, M., Puketza, N., Olsson, R.A., & Mukherjee, B. (1995) Simulating Concurrent Intrusions for Testing Intrusion Detection Systems:Parallelizing. In *Proceedings of the 18th NISSC.* pp. 173-183.

[2] Cramer, M., *et al.* (1995). New Methods of Intrusion Detection using Control-Loop Measurement. In *Proceedings of the Technology in Information Security Conference (TISC) '95.* pp. 1-10.

[3] Debar, H., Becke, M., & Siboni, D. (1992). A Neural Network Component for an Intrusion Detection System. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy.*

[4] Debar, H. & Dorizzi, B. (1992). An Application of a Recurrent Network to an Intrusion Detection System. In *Proceedings of the International Joint Conference on Neural Networks.* pp. (II)478-483.

[5] Denault, M., Gritzalis, D., Karagiannis, D., and Spirakis, P. (1994). Intrusion Detection: Approach and Performance Issues of the SECURENET System. In *Computers and Security Vol. 13, No. 6, pp. 495-507*

[6] Denning, Dorothy. (February, 1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering, Vol. SE-13, No. 2.*

[7] Fox, Kevin L., Henning, Rhonda R., and Reed, Jonathan H. (1990). A Neural Network Approach Towards Intrusion Detection. In *Proceedings of the 13th National Computer Security Conference.*

[8] Frank, Jeremy. (1994). Artificial Intelligence and Intrusion Detection: Current and Future Directions. In *Proceedings of the 17th National Computer Security Conference.*

[9] Ghost, A.K., *et al.* (September 27, 1997). "Detecting Anomalous and Unknown Intrusions Against Programs in Real-Time". DARPA SBIR Phase I Final Report. Reliable Software Technologies.

[10] Hammerstrom, Dan. (June, 1993). Neural Networks At Work. *IEEE Spectrum*. pp. 26-53.

[11] Helman, P. and Liepins, G., (1993). Statistical foundations of audit trail analysis for the detection of computer misuse, *IEEE Trans. on Software Engineering*, 19(9):886-901.

[12] Ilgun, K. (1993). USTAT: A Real-time Intrusion Detection System for UNIX. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*. pp. 16-28.

[13] Kohonen, T. (1995) *Self-Organizing Maps.* Berlin: Springer.

[14] Mukherjee, B., Heberlein, L.T., Levitt, K.N. (May/June, 1994). Network Intrusion Detection. *IEEE Network*. pp. 28-42.

[15] Porras, P. & Neumann, P. (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the 20th NISSC.*

[16] Puketza, N., Chung, M., Olsson, R.A. & Mukherjee, B. (September/October, 1997). A Software Platform for Testing Intrusion Detection Systems. *IEEE Software, Vol. 14, No. 5*

[17] Ryan, J., Lin, M., and Miikkulainen, R. (1997). Intrusion Detection with Neural Networks. *AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop (Providence, Rhode Island),* pp. 72-79. Menlo Park, CA: AAAI.

[18] Sebring, M., Shellhouse, E., Hanna, M. & Whitehurst, R. (1988) Expert Systems in Intrusion Detection: A Case Study. In *Proceedings of the 11th National Computer Security Conference.*

[19] Staniford-Chen, S. (1995, May 7). Using Thumbprints to Trace Intruders. UC Davis.

[20] Tan, K. (1995). The Application of Neural Networks to UNIX Computer Security. In *Proceedings of the IEEE International Conference on Neural Networks, Vol.1* pp. 476-481.

[21] Tan, K.M.C & Collie, B.S. (1997). Detection and Classification of TCP/IP Network Services. In *Proceedings of the Computer Security Applications Conference*. pp. 99-107.