Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks

James Cannady
Georgia Tech Information Security Center
Georgia Institute of Technology
Atlanta, GA 30332-0832

james.cannady@gtri.gatech.edu

Abstract

The timely and accurate detection of computer and network system intrusions has always been an elusive goal for system administrators and information security researchers. Existing intrusion detection approaches require either manual coding of new attacks in expert systems or the complete retraining of a neural network to improve analysis or learn new attacks. This paper presents a new approach to applying adaptive neural networks to intrusion detection that is capable of autonomously learning new attacks rapidly through the use of a modified reinforcement learning method that uses feedback from the protected system. The approach has been demonstrated to be extremely effective in learning new attacks, detecting previously learned attacks in a network data stream, and in autonomously improving its analysis over time using feedback from the protected system.

Keywords: Intrusion detection, neural networks, reinforcement learning.

Introduction

The increasing reliance on computer networks by companies and government agencies means that the importance of protecting these systems from attack is critical. A single intrusion of a computer network can result in the loss, 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. The individual creativity of attackers, the wide range of computer hardware and operating systems, and the ever-changing nature of the overall threat to targeted systems have contributed to the difficulty in effectively identifying intrusions. There are currently two primary approaches to detecting intrusions. Anomaly detection involves identifying activities that vary from established patterns for users, or groups of users. The technique typically involves the creation of knowledge bases that contain the profiles of the monitored activities. The second general approach to intrusion detection, misuse detection, involves the comparison of a user's activities with the known behaviors of attackers attempting to penetrate a system. While anomaly detection typically utilizes threshold monitoring to indicate when a certain established metric has been reached, misuse detection techniques frequently utilize a rule-based expert systems. When applied to misuse detection, the rules become scenarios for network attacks. Unfortunately, since expert systems have no capability for autonomous learning they require frequent updates by a system administrator to remain current. When a new form of attack is identified the signature must be manually encoded as a rule in the expert system for it to be identified in the network stream and these updates may be ignored or performed infrequently by the administrator. Rule-based systems also suffer from a lack of flexibility in the rule-to-audit record representation. Slight variations in an attack sequence can affect the activity-rule comparison to a degree that the attack is not detected by the intrusion detection mechanism.

This paper presents the results of a research effort that investigated the application of an adaptive neural network in the detection of network attacks. Unlike previous applications of neural networks to intrusion detection this research developed a method of autonomously learning new attacks without the need for retraining or manual updates. For the purposes of this research the approach was tested on denial-of-service (DoS) attacks. An attacker carries out a DoS attack by making a computer resource inoperative, by taking up so much of a shared resource that none of the resource is left for other users, or by degrading the resource so that it is less valuable to users. The attacker can flood the system with repeated requests for meaningless processes, continually sending the system garbage data, causing the system to initiate a re-boot, or other similar actions. Each of these attacks reduces or eliminates the availability of computer resources. In a ping flood attack the system machine is rapidly sent ECHO requests by an attacker. The response to each of these requests limits the amount of available system memory for other processes. As the number of successive requests is sent the protected system may slow to a stop as it attempts to manage the increased activity. A similar type of attack, known as a UDP Packet Storm attack, relies on a rapid succession of UDP packets to overwhelm the system. While DoS attacks are less complex than other forms of intrusion they possess characteristics of many other attacks and they are particularly difficult to detect, (National Research Council, 1998). As a result, DoS attacks were used in the evaluations of this approach.

There were four objectives of the research effort described in this paper:

1. ***Determine if an adaptive neural network could autonomously learn to recognize activity that represented a denial of service attack*** – Existing intrusion detection approaches have no independent capability to learn new attacks. The requirement for manual updating of rule-bases and scripts limits the effectiveness of expert systems in a dynamic network environment.

2. ***Determine if an adaptive neural network could accurately identify attacks that had been previously learned*** – While on-line autonomous learning is an important new capability the day-to-day effectiveness of any intrusion detection system is based on the ability of the application to accurately identify attacks in the network data stream.

3. ***Evaluate the ability of an adaptive neural network to recognize new attacks on an initial presentation*** – Existing approaches to intrusion detection have little or no ability to identify new attacks unless the administrator manually updates the rule base.

4. ***Test the ability of an adaptive neural network to refine its analysis of a previously learned attack*** – Existing neural network approaches to intrusion detection demonstrated the ability to accurately identify learned attacks, but they are incapable of on-line modification of their outputs in response to changes in the attacks.

While each of these capabilities provide advances over most existing approaches to intrusion detection, the ability to autonomously learn new attack patterns without manual updating or retraining was the primary focus of the work. The inability of existing systems to autonomously identify new attacks increases the long-term cost of the systems due to the requirement for dedicated personnel to identify and implement the necessary updates. However, more significant is the fact that the lack of autonomous learning by existing approaches results in an intrusion detection system that is only as current as the most recent update and therefore becomes progressively more ineffective over time. To evaluate the ability of an adaptive neural network to satisfy the stated objective a prototype application was developed in C and Matlab™.

The following sections provide an overview of neural networks and previous applications of the technology to intrusion detection. The adaptive neural network that was used in this research effort is then presented with a description of the modified reinforcement learning approach that enhanced the original neural network algorithm. Finally, the results of a series of evaluations of the approach are presented.

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. 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 (Hammerstrom, 1993).

Unlike expert systems, which can provide the user with a definitive answer if the characteristics that are reviewed exactly match those that 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 that it has been trained to recognize. While the probability of a match 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.

Traditional neural networks gain experience initially by training the system to correctly identify pre-selected examples of the problem. The response of the neural network is reviewed and the configuration of the system is refined until the neural network's analysis of the training data reaches a satisfactory level. In addition to the initial training period, the neural network also gains experience over time as it conducts analyses on data related to the problem.

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. Cannady (1998) demonstrated the use of multi-level perceptron/SOM hybrid neural networks for misuse detection in the identification of computer attacks and Bonifacio (1998) demonstrated the use of a neural network in an integrated detection system. Artificial neural networks have also been proposed as alternatives to the statistical analysis component of anomaly detection systems, (Debar, 1992; Frank, 1994; Ryan, 1997; Tan, 1995). Statistical analysis involves 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 (Helman, 1993). 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. Denault (1994) and Fox (1990) proposed neural networks as statistical analysis approaches in the detection of viruses and malicious software in computer networks. The neural network architecture that was selected by Fox was a self-organizing feature map that uses a single layer of neurons to represent knowledge from a particular domain in the form of a geometrically organized feature map. The proposed network 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.

However, each of these neural network-based approaches utilized algorithms that required the complete retraining of the neural networks to learn new attacks. Traditional artificial neural networks (e.g., multi-level perceptron backpropogation systems) were designed to be trained for a specific problem domain with a representative set of data. Once sufficiently trained, these neural networks are capable of accurately analyzing real-world data that matched, or was similar to, the training data. Unfortunately, many application environments are not sufficiently static to be addressed adequately with traditional neural network models. The traditional neural network is unable to improve its analysis of new data until it is taken off-line and retrained using representative data that includes the new information. This process can require substantial time and effort and then result in another "inflexible" neural network that is incapable of effectively analyzing subsequent additional new data over time. Unlike traditional algorithms adaptive neural network architectures are capable of incremental and/or real-time learning in dynamic environments. Adaptive neural networks are an active area of artificial intelligence research that offer the promise of new architectures that are capable of improving their analysis of new data over time without the need for retraining.

CMAC-based On-line Learning Approach

A form of adaptive neural network that has a strong mathematical foundation as well as a variety of successful applications in other problem domains is the Cerebellar Model Articulation Controller (CMAC) neural network (Albus, 1975). The CMAC neural network is a localized three-layer feedforward form of neural network that is designed to produce a series of input-output mappings. CMAC neural networks are used widely in neural network-based control applications because of their capability for on-line learning.

The CMAC neural network uses two mapping stages to process data (Figure 1). The first stage is a nonlinear transformation that map the input $\mathbf{x} \in \boldsymbol{R}^n$ into a higher dimensional vector $\mathbf{z} \in \{0,1\}^J$ which is a sparse vector where the number of nonzero elements is equal to ***C***, referred to as the generalization size. The second mapping results in the output of the CMAC neural network $\mathbf{y} \in \boldsymbol{R}^n$ through a linear matrix-vector product ***w***, where ***w*** is a L x J matrix of real-valued weights. Inputs to the CMAC neural network that are similar in nature will map to many of the same weights. By mapping to the same weights the resulting output of the neural network will also be similar for the inputs. This process, known as generalization, allows the CMAC neural network to respond in a similar fashion to new inputs that are sufficiently close in the input space to existing patterns. Similarly, if the inputs are sufficiently dissimilar then the resulting outputs will be noticeably different.

The mapping process occurs in a series of three layers. A layer consists of input units that forward the binary outputs to a layer of logical AND units that are sparsely interconnected to a layer of logical OR units. The vector **z** is the output of the OR layer. The inputs provided to the neural network are mapped to exactly ***C***, (the generalization parameter), of the internal weights. A second parameter, the quantization value, specifies the discretization of the input space. Each of the elements in the input vector is provided to a series of sensor units with overlapping receptive fields. Each of the sensors produces a 1 if the input falls within its receptive field and 0 if it does not. The width of the receptive field is of each of the sensors determines the input generalization while the offset of adjacent receptive fields determines the quantization. ***C*** is defined as the ratio of receptive field width to the offset of the receptive field.

The binary units of the individual sensor units are provided to a layer of logical AND units. Each of the AND units receives input from a group of sensors that correspond to distinct input variables. The AND units are divided into ***C*** subsets. The receptive fields of each sensor unit are connected to each of the subsets to provide complete coverage of the input space without overlap. Each input vector excites an AND unit from each of the subsets. This results in a total of ***C*** excited units for each input.

The resulting number of State Space Detectors (logical AND units) required may be extremely large. However, since most problem domains do not involve the entire input space the majority of input vectors would not be used. As a result, the size of the output layers and the associated storage requirements are often reduced in CMAC neural network implementations by randomly connecting the AND unit outputs to a subset of logical OR units. In this situation exactly ***C*** AND units for excited by any input and no more than ***C*** OR units will be excited by any input. This approach to memory preservation was preferred when computer memory was limited, but these limitations are not as great now. As a result, the CMAC neural network used in this research directly connected each of the AND units directly to an OR unit.

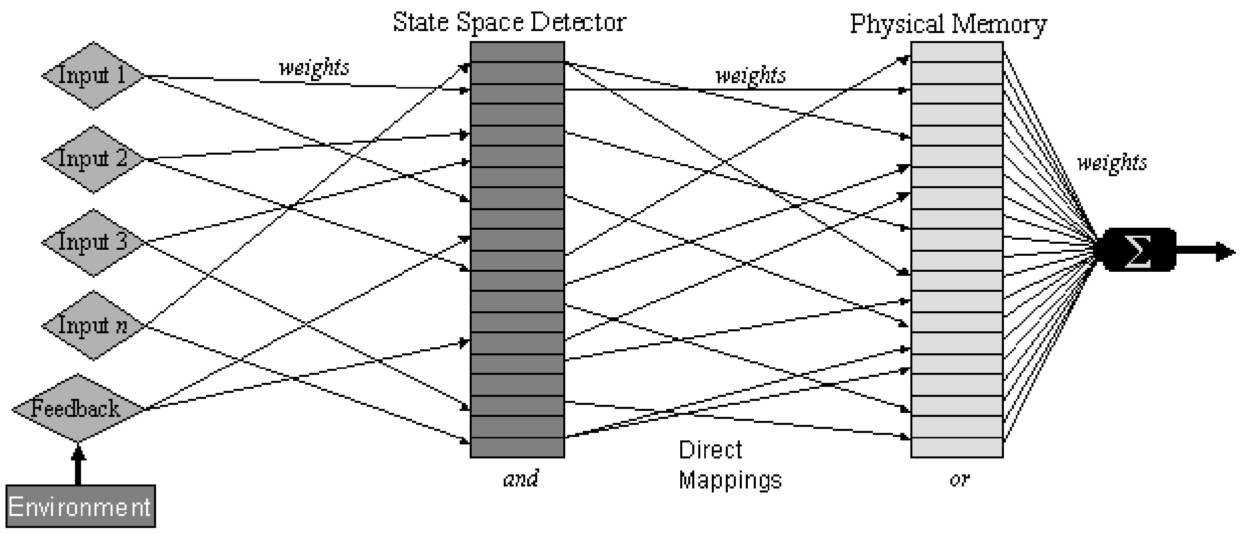

Figure 1: Modified CMAC Architecture

The output of the CMAC neural network is computed by multiplying **z** by the weight matrix of the output layer. The *l*th row of the CMAC weight matrix is adaptively and independently adjusted, traditionally using the least means squared learning algorithm. This results in an approximation of a function $f_l(\mathbf{x})$ that is implemented by the *l*th output unit. While the CMAC neural network possesses excellent local generalization capabilities it is limited in its ability to generalize universally. For this reason CMAC neural networks have been applied most often to real-time robotics and signal processing. Unlike other adaptive neural network the CMAC is able to easily incorporate feedback from the environment. Figure 1 displays the inclusion of feedback from the environment in the CMAC neural network implementation used in this research. This allows the neural network to manage the types of data that are learned. The CMAC algorithm was selected for this approach primarily because of the capability for on-line learning. While the prior work on neural network-based intrusion detection demonstrated the ability to accurately identify network attacks, those efforts were unable to demonstrate the ability to improve analyses without completely retraining the neural network. Like the original CMAC proposed by Albus the neural network that was used in this research utilized a binary kernel function. Traditional least mean square (LMS) learning algorithm was used to update the CMAC weights (w) based on multiplying the difference in the desired output (O_d) and actual output (O_a) by a positive learning factor (b):

$$w_{i+1} = w_i + b\,(O_d - O_a)$$

This CMAC neural network implementation was designed to utilize feedback from the protected system (s) to produce an output that represented the probability of an attack. The feedback from the protected system was in the range 0.0 (system stopped) to 1.0 (system optimal). Each feedback value was based on the combined effect of the input packets on a variety of system state indicators, (e.g., CPU load, available memory, network load, etc.). The output ranged from 0.0 (no attack) to .99 (definite attack) and should be inversely proportional to s. While receiving normal network data the state of the protected system should be nominal, (e.g., 0.75 - 0.99), and the corresponding CMAC output should be small, (e.g., 0.0 – 0.25). However, during a denial of service attack the state of the protected system becomes degraded as the system reacts to the flood of packets and s is reduced. Since the input to the CMAC was limited to the network events and system feedback the LMS learning algorithm was modified to incorporate the feedback by using the inverse of the state of the protected system ($1 - s$) as the desired output from the CMAC:

$$w_{i+1} = w_i + b\,((1\text{-}s) - O_a)$$

The standard LMS learning algorithm was further modified to respond more effectively in a network environment. The inverse of the system state, $(1 - S)$ was used in place of a constant learning factor to increase the flexibility of the learning rate:

$$w_{i+1} = w_i + (1\text{-}S)((1\text{-}S) - O_a)$$

While the typical use of a constant learning factor in LMS algorithm implementations will usually settle to an acceptable error level, the use of a single learning factor prevents the system from varying the rate at which new information is learned. The ability to vary the learning factor offers the advantage of allowing a system to increase or decrease learning rates in response to circumstances in a dynamic environment. By using $(1 - S)$ as the learning factor the CMAC developed in this work was designed to learn faster when the state of the protected system was degraded and at a lower rate when the state was nominal. This results in an adaptive learning rate that responds to the current level of potential threat to the protected system.

Evaluation of Approach

The assessment of the viability of the adaptive neural network approach to intrusion detection was based on four experiments that were conducted through the use of the prototype application:

1. Autonomous Learning of Attacks

The objective of this experiment was to test the ability of the adaptive neural network to learn attacks based on feedback from the environment. In this simulation the environmental input was the state of the protected system. The prototype application included the functionality to simulate a Ping Flood attack against the protected system that could be used to evaluate the autonomous learning ability of the prototype. The number of ping elements was gradually increased in the ten element data vector until the entire vector consisted of the attack elements after sixty iterations. After presenting ten additional "complete" attack vectors the number of attack elements was gradually reduced until the data vectors included only normal/random data elements were included in the simulated data stream. The weights that serve as the "memory" for the CMAC neural network are initialized to random values when the prototype application is started. The prototype is then provided with a complete Ping Flood attack pattern and the resulting initial error is measured as part of the start-up sequence. The prototype was then provided with a gradual simulated Ping Flood attack from one of the established connections. After the presentation of each of the data vectors in the gradual attack the original complete Ping Flood attack pattern is once again presented to the CMAC neural network for analysis. The response error to these subsequent presentations was plotted throughout the learning process, (Figure 2). The difference between the resulting error at each step and the initial error provided an indication of the ability of the adaptive neural network to learn the attack.

After the initial presentation of the complete Ping Flood attack vector the CMAC neural network responded with an error of 97.6758%. This was an extremely high error but not unexpected due to the random nature of the initialized neural network weights. As the CMAC neural network components were trained using the gradual Ping Flood attack vectors and the feedback from the simulated protected system the response error dropped quickly. On the second presentation of the complete Ping Flood attack vector the error was reduced to 2.199%. As the training progressed the response error of the prototype was reduced to 1.94^{-07}%. This extremely small error indicated that the CMAC neural network used in the prototype had accurately learned to recognize a Ping Flood attack in the simulated network data stream.

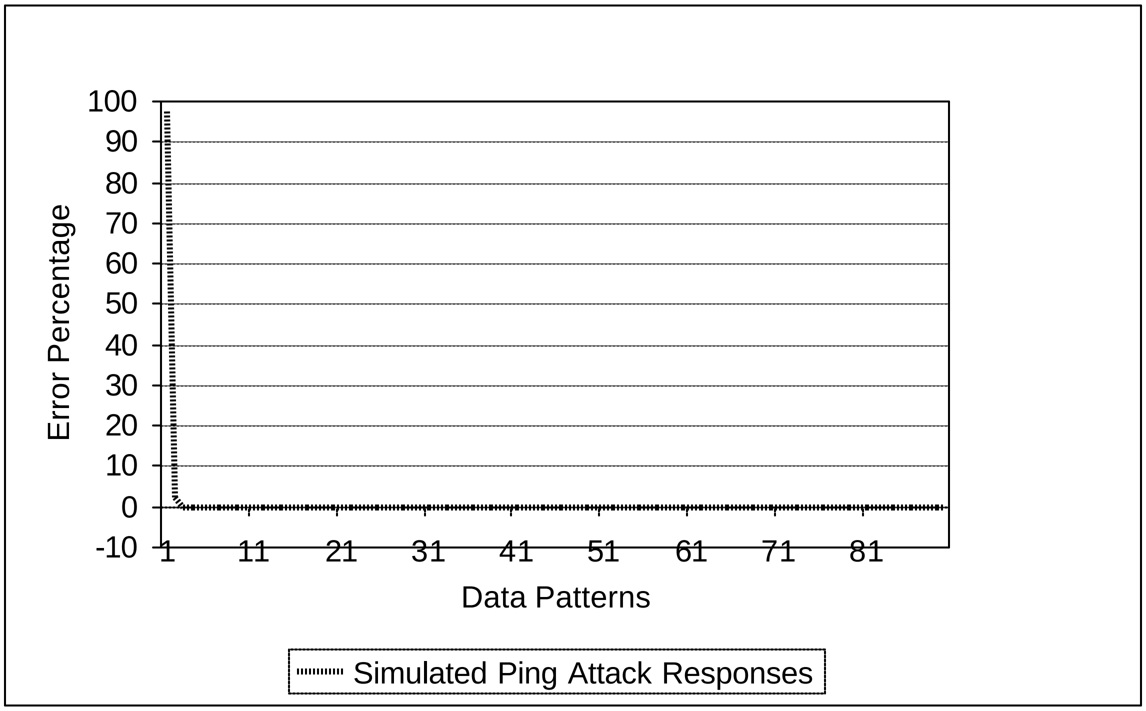

Figure 2 : Response Error of CMAC While Learning Simulated Ping Flood Attack

2. <u>Recognition of New Attacks</u>

The objective of this experiment was to evaluate the ability of the adaptive neural network approach to apply prior learning to the analysis of similar patterns that represent a different attack. One of the most significant advantages of neural networks is the ability to generalize. Generalization allows the neural network to provide a similar response to data patterns that are comparable. The ability to respond effectively when presented with a new attack was a principal goal of the adaptive neural network-based intrusion detection model. The neural network should be capable of applying its prior learning to the analysis of a new attack pattern.

The prototype application included a process that checked the initial response of the CMAC neural network to a UDP Packet Storm attack vector immediately after the CMAC weights were initialized. The initial error of the response was displayed on the prototype user interface at the start of the application. After the CMAC neural network learned the Ping Flood attack in the first experiment the UDP Packet Storm attack vector was presented again to the CMAC neural network to evaluate the ability of the neural network to apply its learning of the Ping Flood attack to another form of denial of service attack. The difference in the initial and post-training responses to the UDP Packet Storm attack vector indicated the ability of the neural network to generalize.

It was hypothesized that the CMAC neural network would be capable of identifying a UDP Packet Storm pattern with a response error less than 15% (the current standard error rate for recognizing existing attacks in commercial intrusion detection systems ((Bonifacio, 1998)) on a presentation of the vector after learning the Ping Flood attack. The initial response error of the CMAC neural network to the UDP Packet Storm attack vector was 93.2869%. This is an extremely high error but it is consistent with the expected response from a neural network with initialized weights. After completing the training the CMAC neural network with the Ping Flood attack in the first experiment the UDP Packet Storm attack vector was presented to the CMAC. The response error of the CMAC neural network on the second presentation was reduced to 2.1992%. This result demonstrated the ability of the neural network to generalize effectively and respond accurately to a similar attack pattern. The extreme accuracy (1.94^{-07}%) in responding to the Ping Flood attack indicated that the CMAC neural network might have suffered from overfitting. This situation occurs when a neural network learns too many input-output examples and memorizes the training data instead of modifying the internal weights to achieve an accurate "curve fit" to the nonlinear data. When a neural network is overtrained it loses the ability to generalize between similar input-output patterns (Haykin, 1999). However, as demonstrated in this experiment the CMAC neural network in the prototype retained the ability to accurately compute an input-output mapping for test data that had not been used in training the neural network. As a result, the CMAC neural network had not suffered from overfitting.

After determining the initial response of the adaptive neural network to the new attack the CMAC neural network learned the new patterns as the new attack progressed. After the presentation of a data vector from the complete simulated UDP Packet Storm attack the original UDP Packet Storm attack pattern is once again provided to the CMAC and evaluated. The response of the CMAC neural network to the complete attack vector was plotted during training, (Figure 3). It was hypothesized that the CMAC neural network would be capable of accurately learning a pattern of activity as a new attack based on the feedback received from the protected system. On the initial presentation of the new attack the CMAC neural network provided an output with a response error of 2.199%. On the second presentation the CMAC neural network response error was reduced to 0.08%. As indicated in Figure 3 the error continued to decline until it reached 8.53^{-14}%.

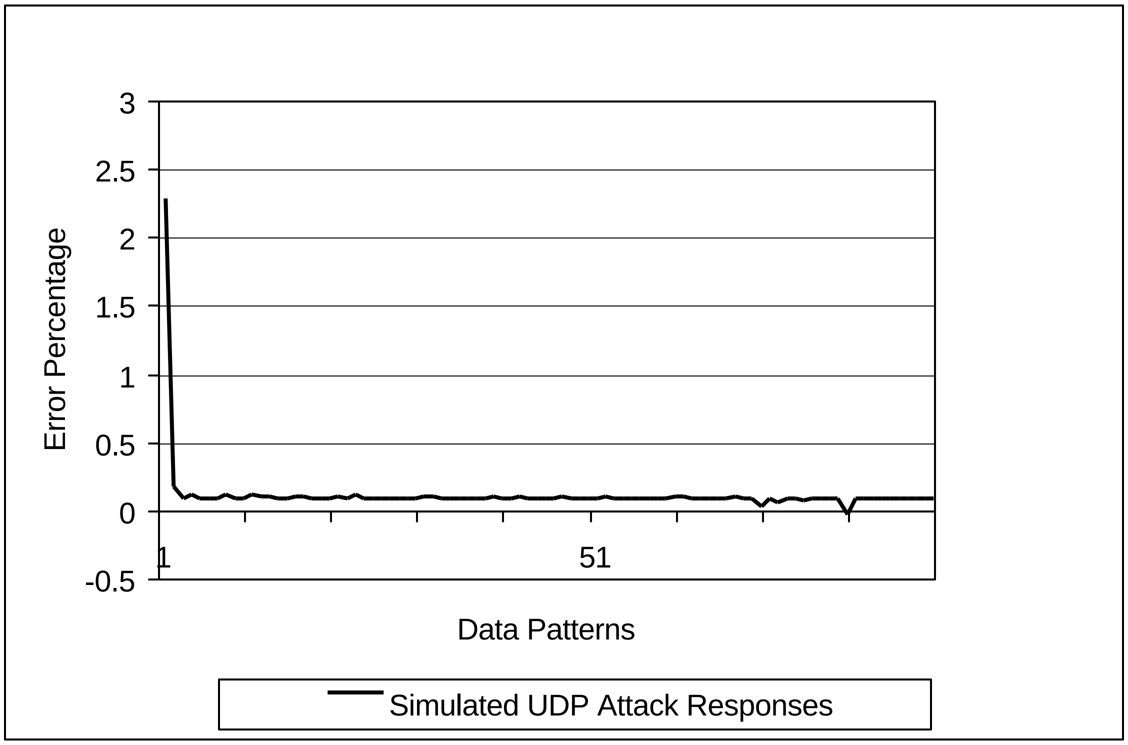

Figure 3 : Error Rate in Learning New Attack Pattern

3. Recollection of Learned Attacks

The objective of this experiment was to evaluate the ability of an adaptive neural network to subsequently identify previously learned attacks from the network data stream. The application was designed to test the ability of the system to quickly and accurately recognize an attack that it has seen before. Some existing adaptive neural network algorithms are capable of learning new information only at the expense of previously learned patterns. This experiment was designed to determine if the CMAC neural network could recall a data pattern subsequent to learning a different pattern. Both the speed and accuracy of an intrusion detection system are important in high-bandwidth network environments.

This experiment was significant since many neural network algorithms are unable to retain prior learning when subsequently trained on new data. As a result, this experiment tested the ability of the CMAC neural network to perform continual learning. To evaluate the ability of the prototype to accurately recall a previously learned attack after learning an additional attack pattern the Ping Flood attack vector was included in the simulated normal data stream at two hundred regular points. The response of the CMAC neural network used in the prototype to each of these presentations was plotted in Figure 4.

The initial response to the Ping Flood attack vector by the prototype was 0.038%. This demonstrated that the CMAC neural network was able to retain memory of the first attack after learning a subsequent attack. However, as indicated in Figure 4, the response error dropped further after subsequent presentations until it stabilized at 3.28^{-05}% after fifty iterations. While the initial recall error was low the fluctuations in the early iterations were probably due to the further refinement of the CMAC weights during subsequent presentations of normal data vectors.

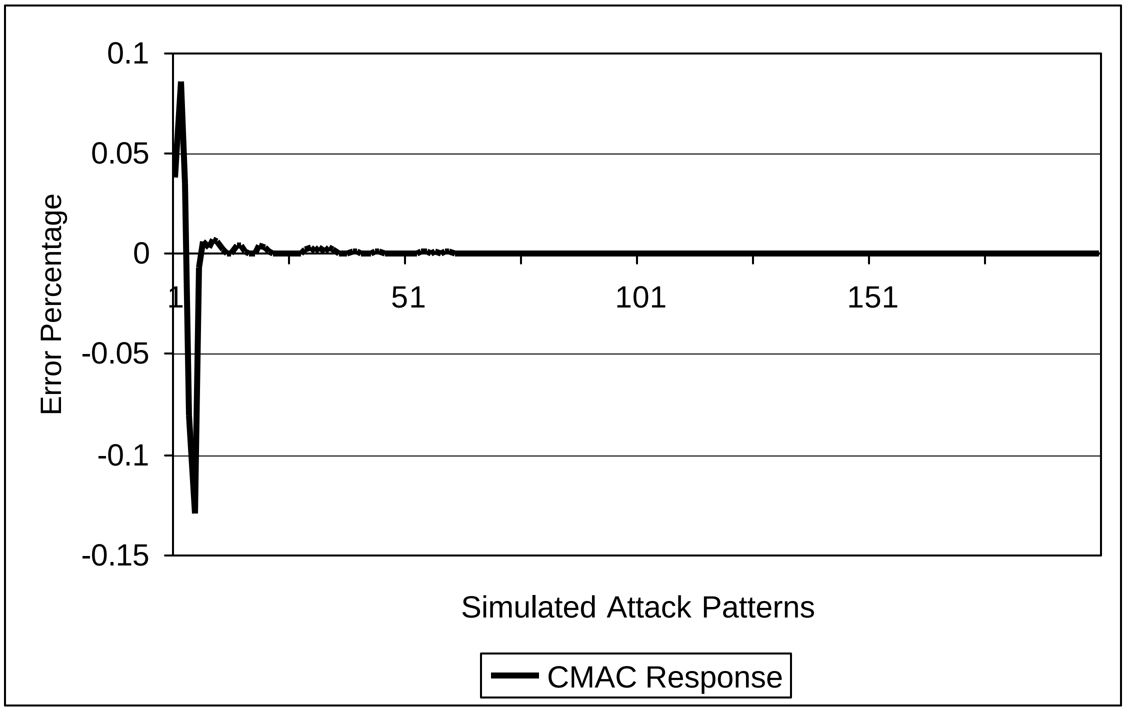

Figure 4 : Response Error During Attack Recall Experiment

4. Refinement of Analysis

The objective of the final experiment was to test the ability to of the adaptive neural network to improve its analysis of new attack during subsequent presentations of the same attack. While the ability of a neural network to learn new attack patterns would be a significant advance over existing approaches to intrusion detection the analysis engine of the intrusion detection system must be able to refine its analysis of an attack based on the feedback from the protected system. As an example, a pattern of activity that is recognized as a 75% chance of being a Ping Flood attack during an initial presentation should have the capability to increase the probability of an attack on a subsequent presentation if the status of the protected system is further degraded.

The prototype application included the functionality to simulate Ping Flood and UDP Packet Storm attacks of increasing severity on the protected system. However, the protected system responds in a consistent manner to the presentation of attack vectors, (i.e., as the number of attack elements increases in the vector the system response is reduced to zero). While this capability facilitates the training of the CMAC neural network used in the prototype in learning attacks an additional form of denial of service attack was used to evaluate the ability of the neural network to refine its analysis in response to feedback from the protected system. In this experiment a data vector containing ten identical elements is presented to the CMAC with the feedback from the protected system. The experiment was conducted after initializing the CMAC neural network weights to zero to avoid conflicting responses with other learned data patterns. The initial feedback from the system is 100%, which represents a protected system state in which all system resources are available. The expected response from the CMAC neural network should be extremely low to represent that the data vector has a low probability of representing a denial of service attack. During each of the fifty presentations of the same data vector the response from the protected system is decreased by two percent until the feedback value was zero. The CMAC neural network should increase its output proportionally to the decrease in the feedback from the protected system to represent an increasing probability of an attack. While this would not be expected to occur in an actual implementation it was designed to demonstrate the ability of the neural network to adapt to changes in the feedback from the protected system that would reinforce the probability of an attack. The results of the CMAC analysis were plotted throughout the on-line learning process. The graph in Figure 5 demonstrates that the CMAC neural network was able to accurately modify its response to the data vector based on the updated feedback from the protected system. Over the fifty iterations of the vector presentation the CMAC neural network responded with an average error of 1.24%.

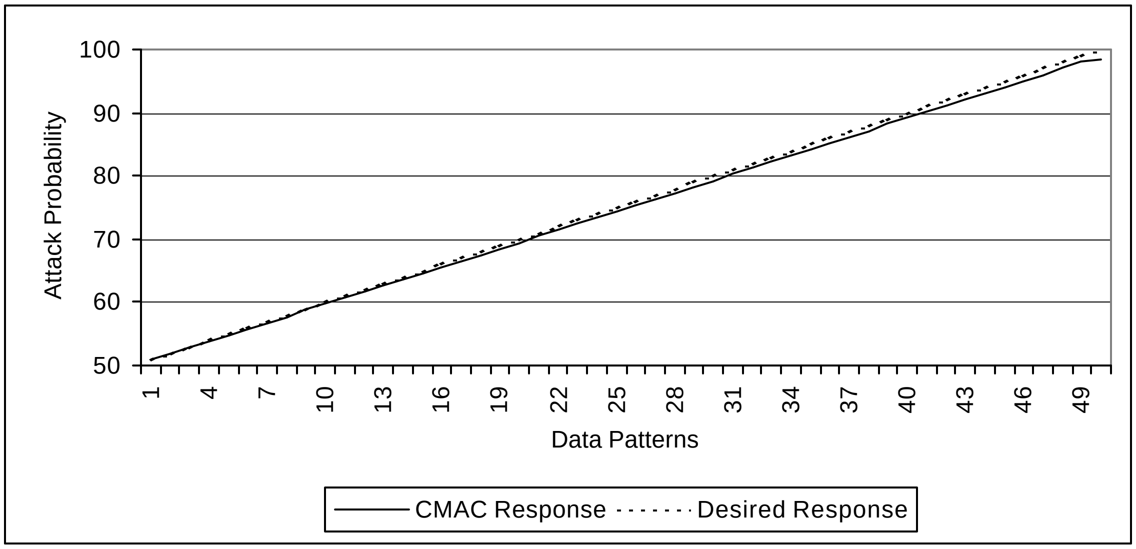

Figure 5 : Results of Evaluation of CMAC Adaptability

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. The results of this research demonstrates the potential for a powerful new analysis component of a complete intrusion detection system that would be capable of identifying *priori* and *a priori* denial of service attack patterns. Based on the results of the tests that were conducted on this approach there were several significant advances in the detection of network attacks:

- On-line learning of attack patterns – The approach has demonstrated the ability to rapidly learn new attack patterns without the complete retraining required in other neural network approaches. This is a significant advantage that could allow the intrusion detection system to continually improve its analytical ability without the requirement for external updates.

- Rapid learning of data – The CMAC was usually able to accurately identify the data vectors after only a single training iteration. This is a significant improvement over other neural network approaches that may require thousands of training iterations to accurately learn patterns of data.

- Extremely accurate in identifying *priori* attack patterns – The use of the modified reinforcement learning approach resulted in an average error of 3.28^{-05}%, compared with an average error of 15% in existing intrusion detection systems. Because other information security components rely on the accurate detection of computer attacks the ability to accurately identify network events could greatly enhance the overall security of computer systems.

- Immediate identification of *a priori* attacks - The approach has demonstrated the ability to effectively identify potential attacks during the initial presentation prior to receiving feedback from the protected system. While the error in the response was higher than during subsequent presentations of the pattern after feedback had been received, the average error rate of 2.199% indicates that the proposed approach has the ability to accurately identify new attacks based on its experience. In addition, the ability of this approach to utilize generalization to provide some indication of attack is an advantage over expert system approaches that require an exact match to coded patterns to provide an alert.

- Ability to autonomously improve analysis – The approach demonstrated the ability to effectively modify its analysis of attacks in real time based on feedback from the protected system. This capability allows the adaptive neural network to continuously improve its analysis without the need for retraining or manual updates.

The results of the tests of this approach shows significant promise, and our future work will involve the application of this approach to other complex forms of attacks which are typically addressed through misuse detection. We are also developing a full-scale integrated intrusion detection and response system that will incorporate the CMAC-based approach as the analytical component.

References

Albus, J.S. (1975, September). A New Approach to Control: The Cerebellar Model Articulation Controller (CMAC). Transactions of the ASME.

Bonifacio, J.M, Cansian, A.M., de Carvalho, A., & Moreira, E. (1998). Neural Networks Applied in Intrusion Detection. In Proceedings of the International Joint Conference on Neural Networks.

Cannady, J. (1998). Applying Neural Networks to Misuse Detection. In Proceedings of the 21st National Information Systems Security Conference.

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.

Denault, M., Gritzalis, D., Karagiannis, D., and Spirakis, P. (1994). Intrusion Detection: Approach and Performance Issues of the SECURENET System. Computers and Security 13 (6), 495-507.

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

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

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

Helman, P., Liepins, G., and Richards, W. (1992). Foundations of Intrusion Detection. In Proceedings of the Fifth Computer Security Foundations Workshop pp. 114-120.

National Research Council. (1998, September). Trust in Cyberspace, 7-8.

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.

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.