Proc. Intelligent Engineering Systems through Artificial Neural Networks ANNIE-2002, St. Louis, MO, vol. 12, ASME Press, New York, NY, 2002, pp. 579-584

NETWORK-BASED INTRUSION DETECTION USING NEURAL NETWORKS

ALAN BIVENS
Computer Science
bivenj@cs.rpi.edu

CHANDRIKA PALAGIRI
Computer Science
palgac@cs.rpi.edu

RASHEDA SMITH
Computer Science
smithr2@cs.rpi.edu

BOLESLAW SZYMANSKI
Computer Science
szymansk@cs.rpi.edu

MARK EMBRECHTS
Decision Sciences & Eng. Systems
embrem@rpi.edu

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ABSTRACT
With the growth of computer networking, electronic commerce, and web services, security of networking systems has become very important. Many companies now rely on web services as a major source of revenue. Computer hacking poses significant problems to these companies, as distributed attacks can render their cyber-storefront inoperable for long periods of time. This happens so often, that an entire area of research, called Intrusion Detection, is devoted to detecting this activity. We show that evidence of many of these attacks can be found by a careful analysis of network data. We also illustrate that neural networks can efficiently detect this activity. We test our systems against denial of service attacks, distributed denial of service attacks, and portscans. In this work, we explore network based intrusion detection using classifying, self-organizing maps for data clustering and MLP neural networks for detection.

INTRODUCTION

Intrusion Detection attempts to detect computer attacks by examining data records observed by processes on the same network. These attacks are typically split into two categories, host-based attacks and network-based attacks. Host-based attack detection routines normally use system call data from an audit-process that tracks all system calls made on behalf of each user on a particular machine. These audit processes usually run on each monitored machine. Network-based attack detection routines typically use network traffic data from a network packet sniffer (e.g., tcpdump). Many computer networks, including the widely accepted Ethernet (IEEE 802.3) network, use a shared medium for communication. Therefore, the packet sniffer only needs to be on the same shared subnet as the monitored machines.

We believe that denial of service and other network-based attacks leave a faint trace of their presence in the network traffic data. Ours is an anomaly detection system that 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 modelled using artificial neural networks (Aussem et al, 2000; Casilari et al, 1998). Therefore we use MLP neural networks to examine network traffic data.

In our system, it becomes necessary to group network traffic together to present it to the neural network. For this purpose, we use self-organizing maps, as they have been shown to be effective in novelty detection (Ypma and Duin, 1997), automated clustering (Niggemann et al, 2001; Vesanto and Alhoniemi, 2000), and visual organization (Kohonen, 1995).

COMPARISON TO OTHER INTRUSION DETECTION SYSTEMS USING NEURAL NETWORKS

There are a few different groups advocating various approaches to using neural networks for intrusion detection. A couple of groups created keyword count based misuse detection systems with neural networks (Lippman and Cunningham, 1999; Ryan et al, 1998). The data that they presented to the neural network consisted of attack-specific keyword counts in network traffic. Such a system is close in spirit to a host-based detection system because it looks at the user actions. In a different approach, researchers created a neural network to analyze program behavior profiles instead of user behavior profiles (Ghosh et al, 99). This method identifies the normal system behavoir of certain programs, and compares it to the current system behavior. Cannady (1998), developed a network-based neural network detection system in which packet-level network data was retrieved from a database and then classified according to nine packet characteristics and presented to a neural network. This method is different from ours because Cannady proposed detection on a packet level, whereas we use a time-window method. Our method allows us to generalize input further than Cannady's method, enabling us to recognize longer multi-packet attacks. In addition, because we are modelling the network traffic in our preprocessing steps, we only need to look at three packet characteristics to identify aggregate trends. Self-Organizing Maps (SOMs) have also been used as anomaly intrusion detectors (Girardin and Brodbeck, 1998). In that work, a SOM was used to cluster and then graphically display the network data for the user to determine which clusters contained attacks.

SYSTEM DESCRIPTION

Our system is a modular network-based intrusion detection system that analyzes tcpdump data to develop windowed traffic intensity trends. In our learning approach, some of our components need time to be trained on the traffic intensity before detection is possible. We call tjis phase an architectural learning period, because during this time we select machine ports for monitoring, determine the structure of the NN, train the SOM, and train the NN. Once these steps have been taken, the system can stop the learning phase and begin detection. The system flow is depicted in Fig. 1. The system reads in tcpdump data and sends it first to the preprocessing module which keeps statistics of the traffic intensity on a source by source basis in each time interval. At any given point, there can be many sources in communication with the

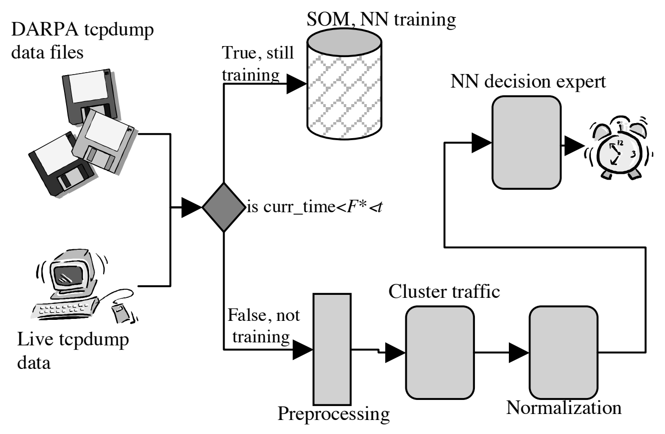
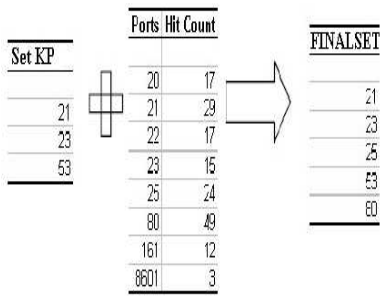


Figure 1: System Architecture and Data Flow Diagram

victim. Therefore, simply grouping the information by sources will not create a uniform representation of data for the neural network. To address this issue, we send the preprocessed source information, which contains the traffic intensity from a source to the target machine, to a clustering module which groups sources with similar traffic intensity trends together during the training phase. The number of thus created clusters is constant and established in the architectural learning phase. When a source is assigned to a particular cluster, the source's traffic intensity is simply combined with the traffic intensity of that cluster. This is done at each time interval, allowing a source to be assigned to different clusters in different time intervals. This type of behavior-based clustering is desired because a machine may be used as an attack source in one time interval and as a normal source in the next one. Once the traffic intensity is grouped into clusters, the group totals are first sent to the normalization module for formatting and then to the actual neural network to render a decision as to the likelihood of a pending attack.

THE LEARNING PHASE AND NEURAL NETWORK STRUCTURE

Prior to collecting and monitoring the network traffic trends, which we believe holds the information revealing intrusions, we must first determine the neural network structure. We present the current intensity of network traffic to the neural network in the form of the number of times a host is accessed through its' different serivces across the network in a certain interval d*t*. To allow a machine to differentiate one application's traffic from another, hosts have up to 2^{16} or 65,536 ports to which or from which traffic travels (Stevens, 94). Monitoring all of these ports through a neural network is unnecessary. It is highly unlikely that all ports on a particular host are used at the same time. Therefore, we must establish which ports are important. To determine the important ports to monitor and thus determine the architecture of our neural network, we introduced an architectural learning phase. We first establish an architectural multiplier *F,* which is multiplied by the time interval d*t* to develop the length of our architectural learning phase. We then observe the network traffic intensity for *F * dt* period of time, cataloging the number of times sources



Set KP
21
23
53

Ports	Hit Count
20	17
21	29
22	17
23	15
25	24
80	49
161	12
8601	3

FINALSET
21
23
25
53
80

Figure 2: Combining given ports with active ones to find the final set of ports

access different ports on the target machine. The number of packets received at the target machine in this period of time form a set *A*. The systems administrator defines for our architectural learning phase a list of known ports to watch (the set *KP*) and how many extra ports (*ep*) the algorithm can choose to add to the list. Our system simply uses the ports given by the administrator *KP* and the top *ep* ports from the remaining most active ones in the observed traffic. The final set of ports used is *FINALSET = KP υ max(ep, A)*. The process of determining the final set of ports to use is illustrated in Fig. 2 (the administrator requests the addition of two ports to *FINALSET*). Once we have determined the ports to monitor, the neural network structure can be established as having *N* * *M* input nodes in which *N* is the number of sources and *M* is the number of monitored ports (|*FINALSET*|).

The following section will show the reason why the sources are clustered for input to the neural network. The first *M* nodes of the neural network input layer represent the total number of packets sent from the first source to the corresponding monitored port. The next *M* nodes of the input layer receive the respective total numbers of packets for the second source in the same order as the first layer, and so forth. An example of this architecture for four sources is given in Fig. 3. In this example, the ports are the same as those chosen in Fig. 2. If only three sources have communicated with the target machine, then the architecture would contain three sets of M nodes in the input layer.

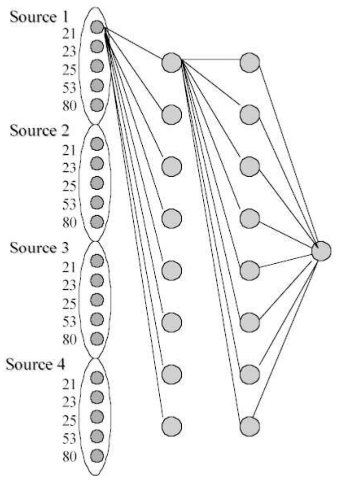


Figure 3: Sample configuration of the neural network with four sources.

Once the neural network is created, the number of inputs it receives is fixed by its structure. As mentioned before, the sources that we should monitor may change frequently from interval to interval. At any given interval, the observed source activity could rise far above or sink far below the activity level of *N* sources used by the neural network in the previous interval. To accommodate such changes, we cluster the sources based on time-windowed behavior and present the cluster totals to the neural network. Once decided, the number of clusters remains the same.

CLUSTERING TRAFFIC TRENDS

Clustering network traffic has been shown to be an effective way of classifying similar trends (Portnoy et al, 2001; Niggemann et al, 2001). To provide both a visual representation of the traffic intensity as well as a meaningful clustering technique, we use a SOM. Discussing the theory of self-organizing maps is beyond the scope of this paper; however, the details of our automated clustering implementation are provided below. Our SOM contains a grid of neurons each possessing a weight vector of the length of |*FINALSET*|. After training, the weight vectors essentially reflect the number of hits to a particular port in a certain time interval *dt*. When determining the best match between an input vector x = $[\xi_1,\xi_2,\ldots,\xi_n]$ and the weight vectors $m_i = [\mu_{i1},\mu_{i2},\ldots,\mu_{in}]$, we use a simple Euclidean distance formula. Once the best match is found, the weights of the neurons are updated through standard SOM formulas with linearly decreasing learning and neighborhood functions (Kohonen 1995).

To develop the clusters from the SOM, we compute a frequency value β to count how many times a particular neuron and members of its neighborhood were chosen as the Best Matching Unit (BMU) during training. The neurons with the highest frequency value (β), are selected to be centroids, the centers of clusters. Our calculation of β is shown in Eq. 1.

$$\beta_{i,j} = freq_{i,j} + \sum_{x=1}^{reach}\left(\frac{freq_{i-x,j} + freq_{i,j-x} + freq_{i+x,j} + freq_{i,j+x}}{1+x}\right), \quad \textbf{(1)}$$

for all valid neurons, where:

- $freq_{i,j}$ = number of times the neuron at position (i,j) was the BMU
- *reach* = the spacing we enforce between cluster centroids

If our algorithm is given the number of clusters to generate (z), it simply chooses the z neurons with the highest β and not adjacent to current centroids to be centroids of the z clusters. In this context, the neuron at position (x_1,y_1) is adjacent to the neuron at position (x_2,y_2) if $(x_2$ - reach <= x_1 <= x_2 + reach) and $(y_2$ - reach <= y_1 <= y_2 + reach). We have an algorithm for computing the number of clusters if it is not given a priori, but will not explain it here due to space limitations. Developing the clusters is only done during the architectural learning phase. After the learning phase, to determine to what cluster a source belongs in a particular time interval, we simply compute the BMU for the source's traffic intensity and select the cluster which has the closest centroid.

TEST RESULTS

As pictured in Fig. 1, our system is built to run on either historical tcpdump binaries or from a real-time tcpdump process. For testing purposes, we ran our system using tcpdump binaries from the DARPA 1999 training dataset (Lippman et al, 2000). We had a great deal of trouble trying to get our neural network to detect all types of attacks simultaneously. However, the neural network performed well when tested on individual types of attacks one at a time. Our training and testing in this area was limited however, because our dataset did not contain many instances of the same attack. Table 1 shows some of our results where sshprocesstable is the name of a particular type of denial of service attack. In Table 1, columns 2, 3, 4, and 5 respectively refer to the correct prediction of normal traffic intensity, the incorrect prediction of normal traffic intensity, the correct prediction of attack traffic intensity, and the incorrect prediction of attack traffic intensity. Testing of additional types of attacks are in progress.

	Correct Normal Predictions	False Negatives	Correct Attack Predictions	False Positives
Union of All Attacks	100%	0%	24%	76%
Sshprocesstable	100%	0%	100%	0%

Table 1: Current Results

CONCLUSION

Many methods have been employed for intrusion detection. However, modeling networking traffic for a simple representation to a neural network shows great promise, especially on an individual attack basis. Also, using SOMs as a clustering method for MLP neural networks is an efficient way of creating uniform, grouped input for detection when a dynamic number of inputs are present. Once trained, the neural network can make decisions quickly, facilitating real-time detection. Neural Networks using both supervised and unsupervised learning have many advantages in analyzing network traffic and will be a continuing area of our research.

REFERENCES

Aussem, A., Mahul, A., and Marie, R., 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, pp. 392-397.

Cannady, J., 1998, "Artificial Neural Networks for Misuse Detection," Proceedings, National Information Systems Security Conference (NISSC'98), October, Arlington, VA, pp. 443-456.

Casilari, E., Alfaro, A., Reyes, A., Diaz-Estrella, A., and Sandoval, F., 1998, "Neural modelling of Ethernet traffic over ATM networks," EANN '98, June, Gibraltar, pp. 304-307.

Ghosh, A., Schwartzbard, A., and Shatz, M., 1999, "Learning Program Behavior Profiles for Intrusion Detection," in Proceedings First USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, April 1999.

Girardin, L., and Brodbeck, D., 1998, "A Visual Approach or Monitoring Logs," In Proceedings of the 12th System Administration Conference (LISA '98), Boston, MA, December, pp. 299-308.

Kohonen, T, 1995, "Self-Organizing Maps," Springer Series, Springer-Verlang Berlin.

Lippmann, R., and Cunningham, R., 1999, "Improving Intrusion Detection performance using Keyword selection and Neural Networks, " RAID Proceedings, Sept, West Lafayette, Indiana.

Lippman, R., Haines, J.., Fried, D., Korba, J., and Das, K., 2000, "The 1999 DARPA off-line intrusion detection evaluation," Computer Networks, 34, pp. 579-595.

Niggemann, O., Stein, B., and Tölle, J., 2001, "Visualization of Traffic Structures," IEEE International Conference on Communications, ICC 2001, vol. 5, pp. 1516 -1521.

Portnoy L., Eskin E., and Stolfo S. J., 2001, "Intrusion Detection with Unlabeled Data using Clustering," In Proceedings of ACM CSS (DMSA-2001), Philadelphia, PA, Nov 5-8.

Ryan, J., Lin, M., and Mikkulainen,R., 1998, "Intrusion Detection with Neural Networks," Advances in Neural Information Processing Systems, vol. 10, MIT Press.

Stevens, R., 1994, "TCP/IP Illustrated Volume 1: The protocols," Addison-Wesley Publishing Company, Reading, Massachusetts, vol. 1, pp. 12.

Vesanto, J., and Alhoniemi, E., 2000, "Clustering of the selforganizing map," IEEE Transactions on Neural Networks, vol. 11, issue 3, pp 586-600.

Ypma, A., and Duin, R., 1997, "Novelty Detection using Self-Organizing Maps," Progress in Connectionist-Based Information Systems, vol. 2, pp 1322-1325.