NETWORK INTRUSION DETECTION SYSTEM USING ATTACK BEHAVIOR CLASSIFICATION

Omar Al-Jarrah
Department of Computer Engineering
Jordan University of Science and Technology
Irbid 22110, Jordan
aljarrah@just.edu.jo

Ahmad Arafat
Systems Engineer, Fortinet
United Arab Emirates
ahmad_arafat78@yahoo.com

Abstract— **Intrusion Detection Systems (IDS) have become a necessity in computer security systems because of the increase in unauthorized accesses and attacks. Intrusion Detection is a major component in computer security systems that can be classified as Host-based Intrusion Detection System (HIDS), which protects a certain host or system and Network-based Intrusion detection system (NIDS), which protects a network of hosts and systems. This paper addresses Probes attacks or reconnaissance attacks, which try to collect any possible relevant information in the network. Network probe attacks have two types: Host Sweep and Port Scan attacks. Host Sweep attacks determine the hosts that exist in the network, while port scan attacks determine the available services that exist in the network. This paper uses an intelligent system to maximize the recognition rate of network attacks by embedding the temporal behavior of the attacks into a TDNN neural network structure. The proposed system consists of five modules: packet capture engine, preprocessor, pattern recognition, classification, and monitoring and alert module. We have tested the system in a real environment where it has shown good capability in detecting attacks. In addition, the system has been tested using DARPA 1998 dataset with 100% recognition rate. In fact, our system can recognize attacks in a constant time.**

Keywords— Intrusion Detection Systems; Network probe attack; Host sweep; Port scan; TDNN neural network

I. INTRODUCTION

Intrusion or threat can be defined as any deliberate action that attempts unauthorized access, information manipulation, or rendering the system unstable by exploiting the existing vulnerabilities in the system. These vulnerabilities could result from wrong practices like weak security policy or software bugs like buffer overflows [3]. Instability of the system is mainly caused by the Denial of Service (DOS) attacks, which attempt to render certain service(s), such as, mail or web service, unstable or to allow information disclosure. Intruders might be system users with restricted privileges who want to gain higher privileges, or might be internet users who want to compromise the system or access sensitive information [1].

Intrusion Detection Systems (IDSs) are used to detect attacks against hosts or networks. There are two types of IDSs: Host-based Intrusion Detection System (HIDS) and Network-based Intrusion Detection System. HIDS protects a certain host or system, while NIDS protects a network of hosts and systems [4].

Despite the fact that many researchers attempted to classify the computer threats and attacks into suitable categories, there is still no adopted standard threats and attacks classification [9] [10] [11]. The most widely used classification for attacks in the research communities is the one adopted by K. Kendall [21]. This classification categorizes the computer attacks into:

- DOS Attacks: DOS attacks try to render the system or certain service unstable.

- User to Root: it tries to gain the root or admin privilege from normal user privilege.

- Remote to User: it tries to gain local account privilege for unauthorized entity.

- Probes: In this type of attacks, the attacker tries to collect any possible relevant information in the network. In this class, two famous types of attacks exist: Host Sweep and Port Scan Attacks. Host Sweep attacks determine the hosts that exist in the network, while port scan attacks determine the available services that exist in the network.

The most well-known classification for IDSs is based on the Approach of Intrusion Detection. There are two common approaches, Anomaly detection and Misuse detection [12]. Anomaly Detection works on two phases: learning phase and detection phase. In the learning phase, it builds a system profile for the normal system parameters and states. While in the detection phase, the system compares the current system parameters with ones in the stored normal system profile, if they deviate in some way, an alert of intrusion is reported to the administrator or an appropriate action could be taken[1][3][4]. The most common advantage of Anomaly Detection is the ability to detect new and novel attacks.

Many Intrusion Detection Systems have been implemented since the concept of Intrusion Detection has appeared. One of the most important and open source NIDSs is SNORT [2]. SNORT is a signature based IDS in which the signatures are expressed through rules, so it is a rule-based IDS. Its performance is high when detecting content-based attacks. Its shortage appears in discovering new and cyber-attacks. SNORT limits its responsive action to report the attack to an administrator console.

Another NIDS, which uses Anomaly Detection, is Minnesota Intrusion Detection system (MINDS) [5]. MINDS

uses three groups of features: features of individual TCP connection, time-based features, and connection-based features. MINDS is adaptive NIDS that was able to detect intrusive activities which SNORT cannot detect. It uses statistical methods and AI techniques in the detection process like Data Mining.

Some Intrusion Detection Systems model the attacks using state transition diagrams [6]. NetSTAT is a signature detection NIDS which describes the state of the network using different parameters like active connections, state of interactions, and the values of network tables. Transitions between states occur when interesting events happened, and the final state could be safe or compromised.

Others have considered different issues in Intrusion Detection Systems such as took the high speed communication network [7]. These projects present new concepts in Intrusion Detection like zero-copy-based packet capture approach [7], which eliminates multiple copies of the packet to minimize the memory overhead. Other improvement was the enhanced packet filter to maximize the number of non-important dropped packets before entering the intrusion detection engine. Also, the efficient application protocol analysis is augmented into NIDS to detect attacks that are related to specific application protocol [9]. Some researchers were interested in attacking and eluding the NIDS itself [8].

This paper uses neural networks as a data processing technique. It shows that the temporal behavior of the attack can be represented in neural network structure with suitable design aspects. The detection process for the stored signatures will be very effective and fast, and the time taken to detect an attack will be constant. The temporal attack behavior is hardwired in the neural network structure to gain fast response and fast rate of attack detection. The system is characterized by high rate of attacks detection and low rate of false positives.

This system works in a universal IP plan, i.e. we do not design the system for a range of addresses or a class of addresses. The numerical values of features are not important but the relations between inputs are the keys. The system captures packets in real time using packet capture engine, it uses multiprocessing to extract relevant information from packets, neural networks that encode the attack types and their temporal behavior to recognize and identify port scan and host sweep attacks, classification neural networks to classify the attacks, and a monitoring and alert system to alert the system administrator.

The rest of the paper is organized as follows. The second section presents an overview of network probes. The third section describes the proposed system structure and components. Experimental results and discussion are presented in the fourth section. Finally, the paper is concluded in the fifth section.

II. NETWORK PROBES

This section discusses the behavior, importance, and types of network probes. Network probes are very important type of network attacks. They are regarded as the first step to launch most of network attacks [14]. Any network attack consists of three stages that usually occur in sequence: network probe, attacks activity, and footprint clearance. Without network probes none of most attacks can be launched. Network probes are known as reconnaissance attacks [15] because their main objective is to collect information about hosts and network services running in the network.

A. Host Sweep Attacks

Network attacks usually start by performing network discovery, which includes host sweep and port scan attacks. Host sweep attack determines the live hosts in the network; the attacker tries to find a gateway to start with or enter from. So, intruders will be searching for weak points to compromise. Without this attack, the intruder will not be able to launch most of the attacks against network resources. There are three common types of host sweep scan techniques: TCP ECHO, UDP ECHO, and ICMP Sweep [16].

B. Port Scan Attacks

Port scanning tries to discover the running services on the victim machine, or tries to check the availability of certain service on the victim. It is well known that each network application running on the machine has a unique port number that it listens to such as port 80 for web browsing. By finding which services are running, a certain attack can be launched against the discovered service, e.g. a mail bomb attack can be launched against mail service to break it down. Port scan techniques presented in this paper have the following three temporal behaviors related to ports and hosts:

- One host- different ports: The attacker scans different ports on a certain host, which is the typical behavior of the port scan. The order of ports is not important; scanning may be sequential or random.

- Different hosts-one port: The attacker scans multiple hosts at the same time with the same port number. This attack is launched against network of hosts looking for hosts running a certain service like DNS, SMTP, or HTTP. This type of scans is more complex than the standard attack.

- Different hosts-different ports: The attacker scans multiple hosts at the same time and each host is possibly scanned with different port. This is an advanced technique of port scanning that tries to hide its activity by initiating random port scans among random hosts. This attack is the most complicated port scan.

In this paper, we will consider seven port scan attacks including TCP SYN, TCP ACK, TCP SYN|ACK, TCP FIN, TCP NULL, TCP XMAS, and UDP/ICMP Error.

III. NETWORK INTRUSION DETECTION SYSTEM USING ATTACK BEHAVIOR CLASSIFICATION

This section describes different issues regarding the structure, theory, and implementation of the system. This system consists of 5 modules including packet capture engine,
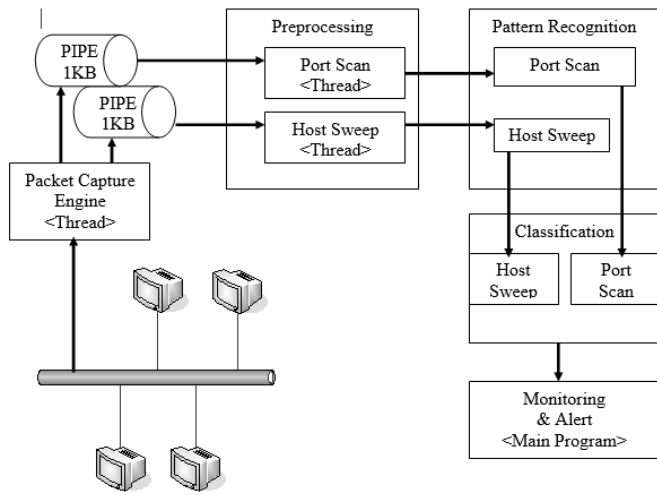
Fig. 1. System block diagram.

preprocessor neural network, pattern recognition neural network, classification neural network, and a monitoring and alert module. The block diagram of the system and the relations between its components are depicted in figure 1.

As shown in figure 1, the system continuously captures the packets on the medium using the packet capture engine module, which is also responsible for extracting relevant features that are required in the subsequent stages. The features are transferred to the next stage using two PIPEs: one for the host sweep and the other for the port scan. The packet capture engine writes the extracted features to the PIPEs which will be read by the next component. The preprocessor reads these features from the PIPEs and process them. The preprocessor consists of two sub components: a host sweep preprocessor and a port scan preprocessor. The function of the preprocessor is to convert the sequential inputs which have a temporal relation to a set of patterns that can be processed simultaneously, i.e. it rearranges the features of the current and previous inputs to make it easier to detect the attacks.

The preprocessing stage collects the features from the packet capture engine stage by reading the features from the read sides of the pipes. As a matter of fact, there are two preprocessors: one for the host sweep and the other for port scan. The features of each packet will be grouped and stored in tapped lines of a Time Delay Neural Network (TDNN).

TDNN is one type of the recurrent neural networks. It has a Tap Line or Tap Delay Line (TDL), which consists of group of taps that orders the temporal inputs. The larger the need for storing past events, the longer the Tap Line [17]. The TDL consists of group of PEs: PE0, PE1...PEn-1, where n is the TDL length.

The output of the preprocessor is directly connected to the pattern recognition component, which is responsible for detecting attacks using neural networks. The pattern recognition consists of two components, one for host sweep and the other for port scan. Because we have different types of attacks, the classification component classifies the detected

attacks in the previous stage into one of the different host sweep and port scan attack types. Finally, this classification is reported to the alert and monitoring module.

The alert and monitoring module is responsible for alerting the system administrator about any existing attacks or threats that exists in the network. The system administrator is responsible for taking the appropriate action.

A. Host Sweep Preprocessor

The host sweep preprocessor module is implemented as a neural network, which consists of the following:

- Input layer: this layer consists of the 10 tapped lines of the TDNN with each tapped line represents one feature in the captured packets.

- Hidden1Syn: this component's design based on the fact that we regard 3 packets out of the last 10 packets is enough to detect an attack. Note that we are interested in detecting attacks based on the relation between current packet and the remaining 9 packets. Since tap number zero represent the current packet, it should be compared with each pair of taps that represent the remaining 9 packets. Therefore, selecting two taps from the remaining 9 taps gives 36 pairs.

    Generally, if we have m out of n packets, the length of the tapped line is n, we can store past n packet features, the number of connection groups that contain m-1 elements will be $\binom{n-1}{m-1}$, and because element number 0 will be connected to each group, this gives us a total of $\binom{n-1}{m-1}$, groups, each has m elements.

- Hidden1Com layer: this layer consists of customized PEs that check for similarities between source IP addresses because the system should investigate whether a triple has the same source address, which is irrelevant of the value of that address. The same argument applies for the destination IP address, the destination port number, and the protocol. The number of the processing elements in this layer is 36 triples * 3 = 108 for each line. Generally, if we have m out of n packets, the total number of connections for this layer is $\left(\binom{n-1}{m-1} * m \right)$.

    The behavior of each line for each triple is as follows:

    - Source line: if the source IP addresses of all members in the triple are same, the output of all triple members is set to 1, otherwise it is set to -1.

    - Destination line: if the destination IP addresses of all members in the triple

different, the output of all triple members is set to -1, otherwise it is set to 1.

- Destination-port line: If the destination port number in all members in the triple is equal to 7 [ECHO port] the output of all triple members is set to 1, otherwise it is set to -1.

- Protocol line: if the protocol of all members in the triple is TCP (Code 6), the output of all triple members is set to 4. If the protocol of all members in the triple is UDP (Code 4), the output of all triple members is set to 2. However, if the protocol of all members in the triple is ICMP (Code 2), the output of all triple members is set to -2. Otherwise, it is set to -4.

- Hidden2Syn: this component consists of a set of modular connections that connect each triple in the hidden1com layer to one PE in the output layer. This reduces the number of PEs in the output layer to 108/3= 36 for each feature. Generally, this layer connects each m-element group to one PE in the output layer that has $\binom{n-1}{m-1}$ PEs.

- Output layer: this layer consists of linear PEs, which produce the average values as follows:

$$Y = \left(\sum_{i=1}^{i=3} X_i \right) / 3 , \qquad (1)$$

where Y is the output and Xi is the input. For example, the protocol line will result in an output of 4 if all inputs are 4 (triple protocol is TCP), 2 if all inputs are 2 (triple protocol is UDP), and -2 if all inputs are -2 (triple protocol is ICMP). The same argument applies to the source line, destination line, and destination port line. Note that each PE in this layer represents the status of the corresponding triple in its line. Generally, the function of this layer is to produce the average value of the inputs.

- OutputSyn: this component connects all outputs from all feature lines to a buffer to be sent to the next module with a total of 36*4 = 144 PE. The 144 outputs are arranged in quadruples. Each quadruple consists of corresponding triple status from each feature line. For example, the first quadruple consists of the first source triple status, the first destination triple status, the first destination port triple status, and the first protocol triple status. Since the number of features that are used in the host sweep is 4, the number of processing elements in this stage can be generalized to $\binom{n-1}{m-1} * 4$.

B. Port Scan Preprocessor

The port scan preprocessor is implemented as a neural network, which consists of the following:

- Input layer: this layer is identical to the input layer in the host sweep processor.

- Hidden1Syn: this component is identical to the Hidden1Syn component in the host sweep preprocessor.

- Hidden1com layer: this layer is similar to the host sweep preprocessor Hidden1com layer with small differences in the output code of the lines. The behavior of the lines in this layer is as follows:

  - Source line: the same as in the host sweep preprocessor.

  - Destination line: the output is set to 1 when all the members of the triple have the same destination IP address. If the destinations are totally different, the output is set to -1. Otherwise the output is set to -4.

  - Destination-port line: the output is set to 1 when all the members of the triple have the same destination port address. If the destination port numbers are totally different, the output is set to -1. Otherwise the output is set to -4.

  - SYN line: the output is set to 1 when all the members of the triple have the same SYN flag. If the SYN flags are totally different, the output is set to -1. Otherwise the output is set to -4.

  - ACK line: the same behavior as the SYN line, but for the ACK flag.

  - FIN line: the same behavior as the SYN line, but for the FIN flag.

  - TCP line: the output is set to 1 when the protocol type in all members in the triple is TCP. If the protocol in all members in the triple is UDP, the output is set to -1. Otherwise, the output is set to -4.

- Hidden2Syn: this layer is identical to the host sweep preprocessor Hidden2Syn component.

- Output layer: this layer is identical to the host sweep preprocessor output layer.

- OutputSyn: this component is identical to the host sweep preprocessor outputSyn component but with the number of PEs in the output is set to 7*36 = 252. The 252 outputs are arranged in groups of 7 elements each. Each group consists of the corresponding triple statuses from each feature line. For example, the first group consists of the first source triple status, the first destination triple status, the destination port triple status, the first SYN triple status, the first ACK triple status, the first FIN triple status, and the first protocol

triple status. Since the number of features that are used in the port scan is 7, the number of processing elements in this stage can be generalized to $\binom{n-1}{m-1} * 7$.

This makes the system portable to any network with any IP plan, since it does not depend on rang of IP addresses or network address to work on.

C. Pattern Recognition Neural Networks

The pattern recognition neural networks use Principal component analysis (PCA) to produce the principle components which differ between host sweep recognition neural network and port scan recognition neural network. PCA is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components [18] [19]. The most common rule that is used in PCA is Sangers which is called Generalized Hebbian Algorithm (GHA) where the principal components are extracted in order with respect to the output unit ordering [20] [26].

For host sweep neural network, the number of principle components is 4. While for port scan, the number of principle components is 7. Note that the number of principle components equals to the number of features, which means that these features are the main components that are needed to recognize these types of attacks. Both of recognition networks, host sweep and port scan recognition networks, take their input from the preprocessor stage.

D. Host Sweep Pattern Recognition Neural Network.

Recall that we identify 4 features or principle components for host sweep recognition including IP source address, destination address, destination port, and protocol type. As the host sweep preprocessor stage has 144 outputs, which should feed the pattern recognition stage, we need 144/4 = 36 recognition subsystem or units to process the 36 quadruples. Figure 3-2 illustrates the host sweep pattern recognition neural network unit. This unit has to be replicated 36 times to formulate the neural network for all inputs. Generally, we need recognition units to process all inputs, and the number of inputs in the unit network will be equals to the number of used features. Since an attack can be in any quadruple, all quadruples are collected in the next stage for complete attack recognition.

The port scan pattern recognition network is still under development and will added in the very near future.
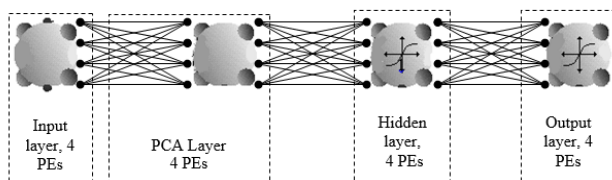

Fig. 1. Host Sweep Pattern Recognition Network Unit.

E. Classification Neural Network

The classification neural network collects the outputs from the host sweep and port scan recognition units, which are 144 for host sweep and 252 for port scan, to produce 4 normalized outputs for host sweep and 7 normalized outputs for port scan. This neural network uses soft max PEs. The overall system consists of two large neural networks, one for host sweep and the other one for port scan. Each is fed from the corresponding pattern recognition units. Generally, the number of processing elements in the election and output layers equals to the number of attacks that the system can recognize.

Host sweep classification neural network consist of the following layers:

- Election layer: this layer consists of 4 PEs which represents the three host sweep attacks. It simply elects the winning PE; the attacks are represented by codes 1 to 4, where 1 represents ping sweep attack, 2 represents TCP ECHO attack, 3 represents UDP ECHO attack, and 4 is reserved for future expansion.

- Output layer: this layer normalizes the output of the previous layer to be in the range of [0 1]. The output value for each processing element should be greater than a threshold to indicate the presence of an attack. This threshold is equal to 0.95.

The election layer and output layer are similar to the corresponding layers in the host sweep. However, each layer consists of 7 PEs. the election stage are coded such as 1 represents TCP SYN port scan Attack, 2 represents TCP ACK port scan attack, 3 represents TCP SYN|ACK port scan attack, 4 represents TCP FIN port scan attack, 5 represents TCP NULL port scan attack, 6 represents TCP XMAS port scan attack, and 7 represents UDP/ICMP Error port scan attack. The training data for these neural networks are the outputs of recognition phase.

IV. RESULTS

In this section, we will show the performance of our system in terms of its capability to detect attacks and its throughput. We have built a test environment to test the performance of our system against SNORT [2]. In addition, we have tested the system using MIT Lincoln Laboratory - DARPA Intrusion Detection Evaluation [24]. Libpcab [21] is used for packet capturing which is implemented using C++ programming language. PIPEs are used for inter-process communication. Neural networks are built and trained using neurosolution package [22].

The test environment is chosen specially to concentrate on the attack recognition capability of attacks and the throughput of the system. It contains an attacker, a victim, and the system. We use NMAP port scanner [23] to generate attacks traffic. The port scan attacks are stored in batch files for each attack. Thus, we have 24 batch files, each perform a specific attack.

In the first test, we compare our system with SNORT [2], which is a rule-based IDS. Both of snort and our system are loaded on the same machine as IDS. The results show that our system can recognize all types of attacks that are described in

this paper regardless of the temporal behavior of the attack. We noted that SNORT missed some attacks like TCP ACK port scan. Since Snort is a rule based system, the augmentation of the rules with new ones is applicable, which may make SNORT capable of recognizing the missed attacks.

The second test is using the standard off-line IDSs evaluation data set of MIT Lincoln Laboratory - DARPA Intrusion Detection Evaluation [24]. MIT has collected a 7-week traffic form a simulation network. Attack traffic was shown in the network on the working hours. The traffic files are stored in the tcpdump [25] format for the 7 weeks. Our system was able to recognize all attacks.

Our tests show that the proposed system can deal with different temporal behaviors of target attacks. These results show a high recognition capability of the proposed system, which is expected since the temporal behaviors of the attacks are embedded in the structure of the preprocessor neural network, i.e. the system is optimized for these attacks patterns. Therefore, the recognition capability will be high in spite of the temporal behaviors of these attacks. DARPA test results show that the proposed system can work with any IP plan with the same efficiency. Our system is characterized by high throughput because after the system is trained, it takes a constant time to detect any attack.

V. CONCLUSIONS

Intrusion detection is an important issue that has received a lot of attention in computer networks. This paper uses TDDNN neural network to recognize the temporal behavior of network attacks. Our system captures packets in real time using a packet capture engine that presents the packets to a preprocessing stage using two pipes. The preprocessing stage extracts the relevant features for port scan and host sweep attacks, stores the features in a tapped line of a TDNN, and produces outputs that represent possible attack behaviors in a pre-specified number of packets. These outputs are used by the pattern recognition neural networks to recognize the attacks, which are classified, by the classifier network to generate attack alerts.

DARPA data sets are used to evaluate the systems in terms of recognition capability and throughput. Test results show that our system detects all types of attacks much faster than rule-based systems such as SNORT.

REFERENCES

[1] Anita K.jones, Robert S. Sielken. Computer System Intrusion Detection: a Survey. University of Virginia, Computer Science Department Technical Report; September 2000

[2] SNORT Network Intrusion Detection System. [Online] [accessed 2006]. Available from URL http://www.snort.org

[3] Sundaram Aurobindo. An Introduction to Intrusion Detection. http://www.acm.org/crossroads/xrds2-4/intrus.html

[4] Przemyslaw kazienko, Piotr Dorosz. Intrusion Detection Systems (IDS) –Part 2- Classification, Methods and Techniques. IT FAQ 2003; 1(4): 17-22.

[5] Minnesota Intrusion Detection System (MINDS). [Online] [accessed 2006]. Available from URL http://www.cs.umn.edu/research/minds

[6] Vigna, Kemmerer. NetSTAT: A Network-Based Intrusion Detection System. Computer Security 1999; 7(1): 37-71

[7] Wu Yang, Bin-Xing Fang, Bo Liu, Hong-Li Zhang. Intrusion Detection System for High Speed Network. Computer Communications 2004 August 15; 27(13): 1288-1294.

[8] Thomas H. Ptacek. Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection. Secure Network Inc; October 2002.

[9] Przemyslaw Kazienko, Piotr Dorosz. Intrusion Detection Systems (IDS) Part I - network intrusions; attack symptoms; IDS tasks; and IDS architecture. [Online] [accessed 2006]. Available from URL http://www.windowsecurity.com/articles/Intrusion_Detection_Systems_ IDS_Part_I__network_intrusions_attack_symptoms_IDS_tasks_and_ID S_architecture.html; 2006.

[10] Dennis Hollingworth. Towards threat, attack, and vulnerability taxonomies. 44th IFIP WG 10.4 Meeting. Monterey, CA, USA; June 25-29, 2003

[11] Faizal Abdollah, Mohd Mas'ud, Shahrin Sahib, Asrul Yaacob, Robiah Yusof, Siti Selamat, Host Based Detection Approach Using Time Based Module for Fast Attack Detection Behavior, Lecture Notes in Electrical Engineering, Volume 157, 2012,   pp 163-171

[12] Rung-Ching Chen, Kai-Fan Cheng, and Chia-Fen Hsieh. Using Rough Set and Support Vector Machine for Network Intrusion Detection. International Journal of Network Security & Its Applications (IJNSA),Vol 1, No 1, April 2009.

[13] Kristopher Kendall. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY; 1999.

[14] Rebecca Cao. RSWATCHER: A PORT SCANNING DETECTOR. Master thesis, Florida State University; 1999

[15] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. IEEE Symposium on Security and Privacy 2004 . Oakland, CA; May 2004.

[16] Whitepaper: Examining port scan methods - Analyzing Audible Techniques. By Dethy, Synnergy Networks Inc; 2001.

[17] Genevieve Orr, Nici Schraudolph, Fred Cummins. Recurrent networks I. Willamette University lecture notes. [Online] [accessed 2006]. Available from URL http://www.willamette.edu/~gorr/classes/cs449/rnn1.html

[18] XLMiner, Data Mining Software online help. [Online] [accessed 2006]. Available   from   URL http://www.resample.com/xlminer/help/PCA/pca_intro.htm

[19] Erkki Oja. Unsupervised learning in neural computation. Theoretical Computer Science 2002; 287:187 – 207.

[20] Jose C. Principe, Neil R. Euliano, W. Curt Lefebvre. Neural and Adaptive Systems: Fundamentals through Simulations. [Online] [accessed   2006]   available   from   URL http://www.neurosolutions.com/products/nsbook

[21] Winpcap: packet capture tool. [Online] [accessed 2006]. Available from URL http://www.winpcap.org

[22] Neuro Solutions: neural network building and training tool. [Online][accessed   2006].   Available   from   URL http://www.neurosolutions.com

[23] NMAP: advanced port scanner. [Online][accessed 2006]. Available from URL http://www.insecure.org/nmap

[24] Lincolin Laboratory, Massachusetts Institute of Technology. [Online] [accessed   2006].   Available   from   URL http://www.ll.mit.edu/IST/ideval/data/data_index.html

[25] TCPDUMP: windows packet capture engine. [Online][accessed 2006]. Available from URL www.tcpdump.org

[26] T.D. Sanger. Optimal Unsupervised Learning in a Single-Layer Neural Network. Neural Networks 1989; 2: 459 – 473.