IJoFCS (2008) 1, 70-74

e-forensic PRESS

The International Journal of
FORENSIC COMPUTER SCIENCE

www.IJoFCS.org

Detecting Attacks to Computer Networks Using a Multi-Layer Perceptron Artificial Neural Network

Dino M. Amaral, Genival Araújo, and Alexandre Romariz

Abstract - In this paper, we present concepts in artificial neural networks (ANN) to help detect intrusion attacks against network computers, and introduce and compare a multi-layer perceptron ANN (MLPANN) with Snort, an open-source tool for intrusion detection systems (IDS). To conduct these comparison experiments, we inserted malicious traffic into the MLPANN to train our ANN, with results indicating that our ANN detected 99% of these input attacks.

Keywords - Networking, TCP/IP, Neural Networking, Multilayer Perceptron, Intrusion Detection Systems.

1 - Introduction

Currently, computer systems are routinely hacked by bots [4], viruses or other harmful entities, as evidenced by much research on attack incidences and their consequences. These attacks cause a loss of billions of dollars worldwide annually. The propagation of malware is always aided by unpatched systems, or users who click on any link that represents their mailbox. In the coming years, we will witness a great explosion of internet access by other devices such as televisions, cell phones and others electronic devices. With internet commerce becoming more common, it is expected that criminal activity will increase.

Detecting attacks against computer systems is an arduous task; it was better studied when Snort [2] was developed and its source code was distributed. Since then, many features of Snort have been improved and other concepts have been incorporated. The neural network (NN), also known as and referred to herein as an artificial neural network (ANN), is a viable alternative to detecting such attacks. In this paper, we report our usage of the multi-layer perceptron algorithm to detect three types of network attacks and Snort to compare among these attacks. [7].

This paper is divided into eight sections as follows: Section 2 covers the basic concepts concerning the three presented attack types; Section 3 explains the multi-layer perceptron, and how we collected data to input into our ANN; Section 4 covers how we trained our ANN; Sections 5 and 6 detail how the algorithm was implemented; Section 7 explains how the results were obtained; and Section 8 presents the conclusion of our research.

2 – Preliminary Concepts

To collect the data that facilitated the training of our NN, we chose three types of attacks:

2.1 – Ping of death

This attack type sends a malicious or malformed packet to a computer. Normally, a ping packet is 64 bytes long. In 1996, attackers began to take advantage of the packet size feature when they discovered that a packet broken down into fragments could add up to more than the allotted 65,536 bytes. Many operating systems were stymied upon receiving an oversized packet, and thus froze, crashed, or rebooted.

Although, sending a packet larger than 65.535 bytes is illegal according the RFC 791[3], this bug is relatively easy to exploit. The fragmentation method is used to send a packet of such large size that when the target computer reassembles this kind of packet, a buffer overflow may occur. This exploitation affects various systems, including UNIX, Linux, Mac, Windows, and routers.

The solution for this situation is to check for each incoming IP fragment, making sure that the sum of the 'Fragment Offset' and 'Total Length' fields in the IP header of each IP fragment is smaller than 65.535 bytes. If the sum is larger, then the fragment is dropped or ignored. This check is performed by some firewalls, to protect hosts that do not have the bug fixed.

2.2 – Brute Force Attacks on Port 22

The secure shell (SSH) service became the most popular to access remote machines. It is easy to use, as there are many clients available to any modern operating system and, above all, the SSH has a strong policy to authenticate users and to protect traffic data. Moreover, SSH is the point used by hackers to gain access to systems with weak policies on passwords, whereby a false assurance is provided if the administrators do not care about this kind of problem.

Despite the fact that SSH allows alternative ways to authenticate remote access, it is possible to use a certificate or simply type in the 'user/password' phrases. Traditionally, the facility chooses and configures the user/password phrase. However, weak passwords are subject to guessing and to brute force attacks. The strength of a password is a function of its length, complexity, and randomness; to avoid selecting a weak password the administrators must work within these constraints. Users tend, however, to choose relatively weak passwords for purposes of easy memorization.

2.3 – SCAN Xmas TREE

A common first step is recognizing the remote operating system. A basic approach to this endeavor uses a port scanner to send malformed packets to target machines. There are various types of scanning, including TCP Scanning, UDP Scanning, ACK Scanning, and e-Window Scanning. According to the answers received, the attacker draws their own conclusion about open ports, kernel versions, operating systems, and other information that is necessary to initiate an attack.

The possible answers include:

- Open or Accepted: The host sent a reply indicating that a service is listening on the port.

- Closed or Denied, or Not Listening: The host sent a reply indicating that connections will be denied to the port.

- Filtered, Dropped or Blocked: There was no reply from the host.

The Xmas Tree Scan was first used in 1999, when the computers could not handle packets with flags FIN, URG and PUSH sets which caused Windows Operating Systems to crash.

3 – Multi-Layer Perceptron

3.1 – Neural Network Concepts

An ANN is a computing architecture inspired from natural biological information processing features.[5][6]

ANNs are made up of processing units connected by communication channels that have multiplicative gains (weights). A signal into an input is multiplied by one weight and added to other signals, the total sum of which is applied to a generally non-linear activation function to calculate the output of each neuron.

The ANN can be used in interconnected layers, and one of its more important properties is its intrinsic learning capacity. During supervised learning, examples of input-output pairs are presented to the network that adapts its internal weights to approximate the desired mapping.

The Multi-Layer Perceptron has at least one 'hidden layer' of neurons, whose outputs are kept isolated and not fed into other neurons. These hidden neurons do not have, in principle a 'desired output' from examples, but rather utilize a well-known optimization technique known as the backpropagation rule, which allows weight adjustments to occur.

No reliable rules exist to determine the number and size of the hidden layers; this is usually accomplished by trial-and-error, with the user trying to find the smallest network that provides optimal performance.

Presentation of examples and weight correction measures through the back-propagation algorithms continue until a stop criterion is attained. Stop criterion usually employ a target value for the mean-square error over the training data.

3.2 – Collecting Data

To train our neural network, the first task is to collect data. An important concern in this phase is to present unbiased data to the network. Data capture was done using tcdump, a packet decoder installed in most Unix and Linux systems. The normalization of these data is necessary to input data to the ANN presented herein, and was accomplished with a dnsi that formats the fields shown in Table 1.

During normalization, the data are collected and we try to input parameters as varied as possible so as not to incur the mistake of letting our network tendentious, and consequently harm the proof concept. Various private IP addresses for classes A (10.0.0.0/8), B (172.17.0.0/16) and C (192.168.0.0/16) were used as input.

The fields used as input for our ANN correspond to the fields in the tcpdump format:protocol, source IP address, source port, destination IP address, destination port, TCP flag, and packet length.

Table 1
Input Data in the Neural Network

Name	Description	Modification
Protocol	Define the protocol : TCP, UDP e ICMP	Without Modifications
Source_IP	Address of source IP	Converted to logarithmic scale
Destination_IP	Address of destination IP	Converted to logarithmic scale
Source_Port	Source Port	Converted to logarithmic scale
Destination_Port	Destination Port	Converted to logarithmic scale
Flags	Flags used by TCP protocol	Without modifications
Length	Length of IP Packet	Converted to logarithmic scale

The Nmap scanner [9] was used to genarate traffic. Source and destination ports were varied randomly.

4– Network Training

Supervised training through the backpropagation algorithm has two main steps:

1 - Data presentation and calculation of network output.

2 - Error calculation and weight changes according to a gradient-descent technique for optimization:

$$\Delta w_{ij} = \frac{\delta E_j}{\delta w_{ij}}$$

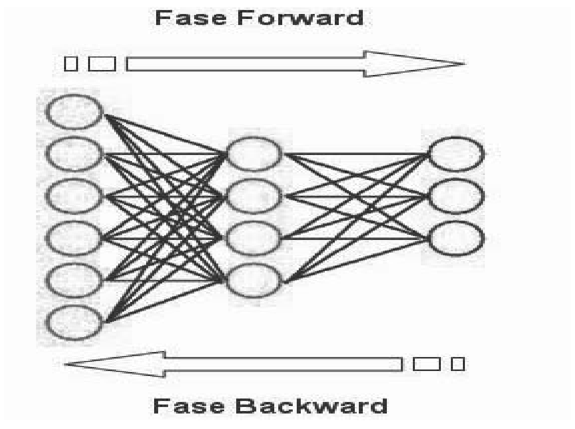

Figure 1: Two steps of backpropagation algorithm

The data were arranged in two sets, as is usual in ANN training. These sets included the training set, containing 7,000 examples of traffic variables, both for normal traffic and during an attack. Each example is composed of seven variables (Table 1) and a desired output (0 for normal traffic, 1 for attack). A second set (the test set) is composed of 5,000 other traffic variable examples that will not be used to adapt the ANN weights, but only to evaluate its performance and generalization capacity.

Initially, the archives were preprocessed for the backpropagation algorithm used in the MLANN . In this way, were created four archives in the Excel format:

- Input Matrix - Training: 7,000 x 7 dimension
- Input Matrix - Validation: 5,000 x 7 dimension

Each input matrix has a desired output (class), and the preparation of these matrices is described hereinafter:

- Output Matrix - Training: the output layer will consist of one node, which is responsible for warning of a network attack (output 1), or if the network traffic is normal (output 0). Thus, the output dimension will be 7,000 x 1.

File output	Neural net output
0	normal traffic
1	net under attack

- Output Matrix - Validation: is the same described procedure for the Output Matrix - Training, described above, but the matrix dimension will be of 5,000 x 1.

After the training of these archives, we will obtain four matrices that will store the prepared values for developing the ANN.

We used the MATLAB programming language to develop the ANN, where the original files, previously described were treated for the ANN's entry.

The logarithmic normalization procedure derives values between 0 and 1. However, because the trained ANN cannot assume this exact value range, we modified the exit matrices range to be between 0.1 and 0.9, whereby an exit value of '1' is changed to 0.9, and that of '0' becomes 0.1. The final treatment of these files was accomplished in Excel.

5 – Development of the Algorithm in Matlab Language

The ANN's architecture was defined as follows:

1 – The Input Layer has seven responsible neurons for receiving the input elements (patterns) of the ANN;

2 – The Output Layer has one neuron, responsible for showing the response of the ANN;

3 – The Hidden Layer was examined initially with three neurons, but simulations were conducted with four neurons.

Given seven input variables, the network output calls for seven input neurons in the first layer, one hidden layer whose size is varied after each training, and one output layer with a single neuron, indicating the network architecture.

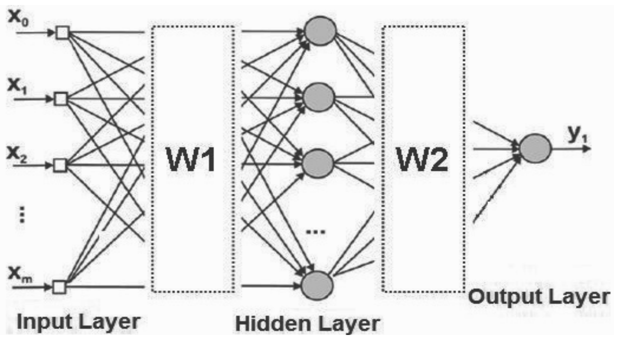

Figure 2: Weights Generated in the Backpropagation Net.

Therefore, we describe the ANN as the design above (Table 2) with m=7 and one neuron in the exit and the Hidden Layer neurons (initially three).

6 – Simulations Results

As stated previously, we initially examined the hidden layer with three neurons obtaining the results presented in tables 2 – 5.

Table 2

Learning rate equal 0.3 and 3 neurons in the hidden layer

Learning rate	Number of epochs	Neurons in the hidden layer	Training error	Validation error
0.3	100	3	0.1889	0.1932
0.3	300	3	0.0906	0.0892
0.3	500	3	0.0311	0.0372
0.3	700	3	0.0869	0.0956

Table 3

Learning rate equal 0.3 and 3 neurons in the hidden layer.

Learning rate	Number of epochs	Neurons in the hidden layer	Training error	Validation error
0.15	100	3	0.1849	0.1892
0.15	300	3	0.0786	0.0824
0.15	500	3	0.0669	0.0676
0.15	700	3	0.0109	0.0168

Table 4

Learning rate equal 0.15 and 3 neurons in the hidden layer.

Learning rate	Number of epochs	Neurons in the hidden layer	Training error	Validation error
0.3	100	4	0.1026	0.1056
0.3	300	4	0.0691	0.0696
0.3	500	4	0.0557	0.0564
0.3	700	4	0.0669	0.0684

Table 5

Learning rate equal 0.3 and 4 neurons in the hidden layer.

Learning rate	Number of epochs	Neurons in the hidden layer	Training error	Validation error
0.15	100	4	0.1066	0.1108
0.15	300	4	0.0766	0.0792
0.15	500	4	0.0686	0.0704
0.15	700	4	0.0683	0.0712

7 – Conclusion

The need for continuous improvements in device security motivates new alternatives for identifying computer network attacks. The use of ANNs for implementing network security is an attractive alternative to other common and less effective anti-attack methods. Based on the signatures of Snort [2], we established malicious packages for assembly of the ANN training files. Our results indicated a 99% success rate for recognizing potential attack code.

Commercial applications require supplementary input parameters, such as a package timestamp and package payload, among others. The diversification of the examples is another factor to consider to improve the training of the ANN. Nonetheless, we consider our

99% success rate to be a highly promising framework for developing future ANNs against malicious traffic.

References

[1] http://www.denunciar.org.br/twiki/bin/view/SaferNet/Noticia20070704035010

[2] http://www.snort.org

[3] http://www.rfc.org

[4] B. Saha and A. Gairola. "Botnet: An Overivew". CERT-In White Paper, CIWP-2005-05, June 2005

[5] Haykin S. - "Redes Neurais Princípios e Práticas", Editora Bookman, 2001.

[6] Braga, A. P., Carvalho A. C. P. L. F. e Ludemir T. B. - "Redes Neurais Artificiais", Editora LTC, 2000.

[7] ROCHA, D. L. - "Utilização de um ambiente de honeynet no treinamento de redes neurais artificiais para detecção de intrusão", ENE-FT UnB, 2006.

[8] http://www.vmware.com

[9] http://www.insecure.org/nmap.