NeuDetect: A Neural Network Data Mining Wireless Network Intrusion Detection System [*]

C.I. Ezeife
School of Computer Science
University of Windsor
Windsor, Ontario N9B 3P4
cezeife@uwindsor.ca

Md. Zillur Rahman
School of Computer Science
University of Windsor
Windsor, Ontario N9B 3P4
woddlab@uwindsor.ca

ABSTRACT

This paper proposes NeuDetect, which applies a classification rule mining Neural Network technique to wireless network packets captured through hardware sensors for purposes of real time detection of anomalous packets. To address the problem of high false alarm rate confronted by current wireless intrusion detection systems, this paper presents a method of applying artificial neural networks mining classification technique to wireless network intrusion detection system. The proposed system, NeuDetect, solution approach is to find normal and anomalous patterns on pre-processed wireless packet records by comparing them with training data using Back-propagation algorithm.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: [Database Applications, Data Mining]; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Invasive software, Unauthorized access*

General Terms

Wireless intrusion detection

Keywords

hardware sensor, wireless attacks, Wi-Fi, neural network, back-propagation algorithm, training data

.

1. INTRODUCTION

An Intrusion Detection System (IDS) is any system that monitors network traffic and is able to detect non-permitted or anomalous deviations (security violations). An IDS can be either host-based for monitoring system calls or logs, or network-based for monitoring the flow of network packets. Wireless Local Area Network (WLAN) is based on IEEE 802.11 standards. WLAN allows for three different ways to configure a network: ad-hoc or Independent Basic Service Set (IBSS), infrastructure or Extended Service Set (ESS), and a mixture of both or Basic Service Set (BSS). In ad-hoc network, connection is established for the duration of one session and requires no base station (computer). A wireless access point (WAP or AP) is a device that connects wireless communication devices together to form a wireless network. When a WAP is introduced in IBSS, the network becomes BSS and the AP acts as a master to control the stations within that BSS. The two authentication mechanisms provided in WLAN are open authentication and shared key authentication, which require clients to provide a secret key to authenticate to an AP by communicating with frames. Communication between stations and AP occur through packets (or frames). A typical wireless packet has the following information for data attributes:
1. Wireless Packet Information (has about 6 sub fields).
2. 802.11 (has about 20 sub fields for attributes as frame control id, source MAC address, service set identifier (SSID), etc.).
3. Encrypted Data (with some sub fields).
4. Raw Data (in coded or encrypted form).
Wireless network attacks belong to five classes, namely:
(1) Access Control Attacks (e.g., WEP attack, war-driving attacks, Rogue Access points, MAC spoofing).
(2) Confidentiality Attacks (e.g., WEP key-cracking, Man-in-the-middle): These attacks intercept private information sent over wireless associations by capturing data frames.
(3) Integrity Attacks (e.g., Replay and 802.11 Frame Injection): These attacks send forged, control, management or data frames over wireless networks to mislead the recipient or facilitate another type of attack (e.g., DoS).
(4) Authentication Attacks (e.g., Application Login Theft): Attacker captures user credentials (e.g., e-mail address and password) from cleartext application protocols like WinSniffer (http://www.winsniffer.com/) installed by attacker through back door.
(5) Availability Attacks (e.g., RF Jamming): These attacks impede delivery of wireless services to legitimate users by e.g., crippling WLAN resources.

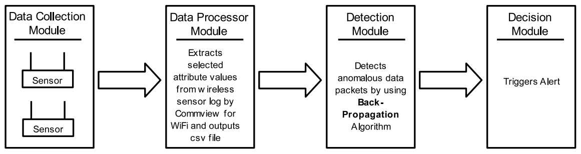
Data Collection Module
Sensor
Sensor
Data Processor Module
Extracts selected attribute values from w ireless sensor log by Commview for WiFi and outputs csv file
Detection Module
Detects anomalous data packets by using Back-Propagation Algorithm
Decision Module
Triggers Alert

Figure 1: The Proposed NeuDetect System Architecture

1.1 Related Work on Data Mining Based Network IDS

In [3], the authors presented an intrusion detection method based on Dynamic Growing Neural Network (DGNN) for wireless networking and considerable challenges remain unexplored. This involves intrusion detection models for wireless networks requiring reliable mechanisms for providing hard-to-get training data in wireless network environment, as well as intrusion detection that has no prior knowledge of relationships between attack types and attributes of the network audit data. The proposed Wireless network intrusion system, NeuDetect, applies Neural Network Back-Propagation classification technique [2] to the important problem of network intrusion detection.

1.2 Contributions and Outline

This paper proposes a Wireless Network Intrusion Detection System (NeuDetect), which builds hard-to-get training dataset for wireless network from crafted attacks, and then classifies incoming data packets by comparing them with the trained attack database. NeuDetect system's scientific solution approach, entails applying Neural Network Back-propagation learning algorithm for classification of training database of wireless attacks for wireless network connection records promptly captured with a set of proprietary Network Chemistry hardware sensors.

Section 2 presents the proposed NeuDetect Wireless Intrusion Detection System. Section 3 presents conclusions and future work.

2. THE PROPOSED NEUDETECT WIRELESS IDS

2.1 Components of the Proposed NeuDetect Intrusion Detection System

The architecture of the proposed wireless network intrusion detection system, NeuDetect, is shown in Figure 1. These components are discussed next.

Data Collection Module

The goal of this module is to capture wireless network packets successfully from a selected Access Point (AP) and log them into the database. Our Packet Capture module contains the proprietary Network Chemistry wireless hardware sensors [4]. Then, the feature extractor module converts the raw data to readable format with help of CommView (CommView for WiFi 2007) for WiFi software and outputs a csv file (csv stands for Comma Separated Values where attributes values are simple text separated by commas). The Data Collection module installs and configures: 1) Wireless Access Point (AP), 2) RF Sensors including both a sensor server and a sensor client software, 3) CommView for WiFi software, 4) MS SQL Server database. Once the sensor is configured, it is able to detect wireless networks.

Data Processor Module

This module extracts feature vector from the network stream and submits the feature vector to the Back-Propagation (BP) model. After a detailed study of network attacks we have selected the following ten feature/ attributes and their descriptions enclosed in brackets: 1. ESSID (the Access Point (AP) Name), 2. SrcMAC (source MAC Address), 3. destMAC (destination MAC Address), 4. SrcIP (the source IP address), 5. destIP (the destination IP address), 6. Packet Size (the number of bytes), 7. Time (time stamp), 8. srcPort (source port number), 9. destPort (destination port number), 10. Channel (channel number). We have used an extraction, transformation and loading (ETL) tool called MS SQL Server Information Services (SSIS) to load from .cvs file from Commview for WiFi into MS SQL Server database.

Detection Module

The function of this module is to analyze the network stream and to draw a conclusion of whether normal or anomalous. To analyze the network stream, it uses Back-Propagation (BP) technique, The next step entails designing the MLP Neural Network to run the Backpropagation algorithm on. The NeuDetect system has one hidden layer, 10 attributes from feature vector as input layer and 1 attribute as output layer. The dimension of the hidden layer is determined as 3 using the formula Hidden layer = $\sqrt{(1 * 10)} \approx 3$. The activation function for the neuron or unit, represents the output of the unit given its net input x, and is the sigmoid function. $f(x) = 1/(1 + e^{-x})$. The weights of different neural units and the bias thresholds are set to a small random number from -1 to 1. By doing so, the convergence of the BP network can be assured.

2.2 The Proposed NeuDetect Algorithms

The BP training algorithm that we have used in our proposed system is given as Algorithm 1.

ALGORITHM 1. *(NeuDetect_Train: Learning Network Weights for Classification)*

```
Algorithm NeuDetect_Train()
Input:   10 feature input units (neurons) for 10 Attributes (I)
         Set of hidden and output layer attribute units (O),
         Target output value, T, weights $w_{ij}$ between,
         biases $\theta_j$ of each output unit j
         input unit i and output unit j, an input packet
         learning rate (l)
Output:  Anomalous Packets (A), Normal Packets (N)
begin
(1) Set all weights $w_{ij}$ to random values ranging from -1.0 to +1.0
(2) Set the input packet to equal the neurons of the net's input layer
(3) Activate each output/hidden neuron j of the forward
neuron by computing its net input $I_j$ from all units i
it preceding and connected to neuron j as:
      3.1 $I_j = \sum_i w_{ij} O_i + \theta_j$
      3.2 Pass the result to an activation function, which computes
      the output value $O_j$ of this neuron as: $O_j = \frac{1}{1+e^{-I_j}}$
      3.3 Compute the error of each output unit $Err_j$ from
      target output $T_j$ as: $Err_j = O_j(1 - O_j)(T_j - O_j)$
      3.3 Compute the error of each hidden unit $Err_k$
      connected to a forward neuron k as:
      $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$
(4) Compare the calculated output pattern $O_j$ to the desired
      target pattern $T_j$ and compute the root mean square
      error value $Err_{rms}$ as: $Err_{rms} = \sqrt{(\sum_{i=1}^{n} Err_i^2)}$
```

```
(5) If the root mean square error $Err_{rms} \approx 0$
        then, algorithm has finished learning and end
        (5.1) Else, change all weight values of each weight matrix
        and biases using the formula:
$w_{ij}$ = weight($w_{ij}$) + learning rate(l) * output error($Err_j$ * output($O_i$)
        $\theta_j$ = bias ($\theta_j$) * learning rate(l) * output error($Err_j$.
(6) Go to step 2
end
```

During detection, the trained MLP network is used to compute the output error, e.g., Err_{10} for a network with output unit of 10. This error is compared with a set threshold to compute an anomaly score. Then, the packet (record) is labeled as either anomalous if this anomalous score is positive, or normal if it is negative or unknown if it is zero. The algorithm for classifying an incoming record as anomalous or not is the NeuDetect Detection algorithm and this is provided as Algorithm 2.

Algorithm 2. *(NeuDetect_Decision: Classify Record Intrusion or Not)*

```
Algorithm NeuDetect_Decision()
Input: Anomaly Score (X), Attack Number (N), threshold(T)
Output: Normal (M), Known Attack (K), Unknown Attack (U)
            from others $\geq$ minimum deviation, V
begin
1. If positive anomaly score $\geq$ threshold T
        1.1 Level as Anomalous and match with attack database
        and trigger alert
        1.2 Else if negative anomaly score
            Skip it as Normal
        1.3 Else if zero or < threshold T
            Level as Unknown Attack
            And send back to training process to reevaluate
end
```

When the detection module estimates that the network stream is anomalous and decision module can not match with any known "Attack", but the threshold of the detection module has been reached, then this module will draw the conclusion that the network stream being detected is "Unknown". Here, the "Unknown" virtually represents a new attack type which the detection module has not been trained to identify. The "Unknown" examples then will be added to the training samples of the BP model, and after training with the "Unknown" examples, the detection module will have the ability to identify it accurately. After such a process, the "Unknown" becomes known "Attack", which implies that the ability of the BP model can be enhanced dynamically. All of these modules together constitute the NeuDetect prototype system.

2.3 Application of NeuDetect Algorithm on Sample Data

Example 1: Assume a 2 layer Multilayer Perceptron Neural Network which has one set of 6 input units (x_1, x_2, x_3, x_4, x_5, x_6) corresponding for example, to a database tuple like the wireless network packet received. The complete NeuDetect Network has 10 feature inputs but to make the example, understandable, we are using only 6 of the 10 inputs in this example. The attributes for our six input x_i are respectively SrcMAC, destMAC, srcPort, destPort, PacketSize and ESSID. The network has one hidden layers with 3 units (x_7, x_8, x_9) before a final output layer x_10. With the

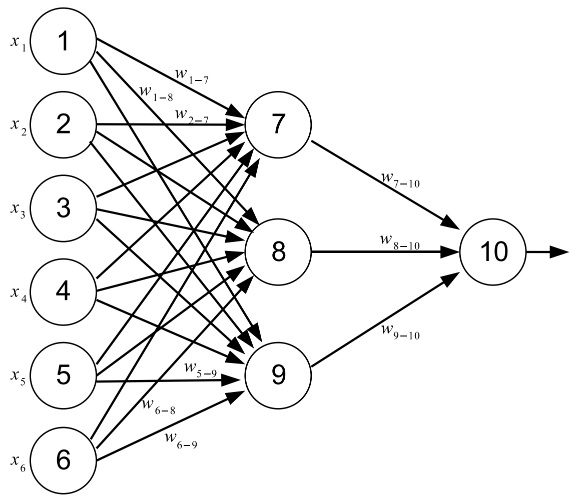

Figure 2: Neural Network for NeuDetect with 6 of 10 inputs, 1 output and 3 hidden layers

values of the 6 input units x_1 to x_6 respectively as: (0.894, 0.134, 0.80, 0.525, 0.11, 0.907), Assume the randomly chosen initially assigned weight (between -1 and 1) connecting each input unit i to a corresponding output unit j is represented as w_{ij} as shown initially below for sample NeuDetect MLP network given as Figure 2. $w_{17} = 0.2$, $w_{18} = -0.3$, $w_{19} = 0.4$, $w_{27} = 0.1$, $w_{28} = -0.5$, $w_{29} = 0.2$, $w_{37} = -0.3$, $w_{38} = -0.2$, $w_{39} = 0.1$, $w_{47} = -0.2$, $w_{48} = 0.3$, $w_{49} = -0.1$, $w_{57} = -0.4$, $w_{58} = 0.3$, $w_{59} = -0.2$, $w_{67} = 0.5$, $w_{68} = -0.3$, $w_{69} = -0.5$, $w_{710} = 0.2$, $w_{810} = 0.4$, $w_{910} = -0.4$. Also, the initial bias values for the four non-input units are $\theta_7 = -0.4$, $\theta_8 = 0.2$, $\theta_9 = 0.5$, $\theta_1 0 = 0.3$. Show the process of using the Backpropagation learning algorithm to find the refined acceptable weights of the network that minimize the squared error between the target output class label T = 1 of the training tuple and the predicted output from the Neural Network.

Solution 1:
Given a unit j in a hidden or output layer, which has a connection from a unit i in the previous layer to unit j, where the output of unit i is O_i and the connection weight between units i and j is w_{ij}, and θ_j is the bias of unit j, the net input, I_j to unit j is obtained with the equation: $I_j = \sum_i w_{ij} O_i + \theta_j$. Each unit j in the hidden and output layer accepts its net input I_j, which it applies to an activation function like the sigmoid function to produce the output of the unit O_j, using the following formula. $O_j = \frac{1}{1+e^{-I_j}}$. The computed net inputs and outputs for hidden layer units 7, 8 and 9 as well as the output unit 10 of the example MLP network of Figure 2 are given below. Input
$I_7 = 0.894 * 0.2 + 0.134 * 0.1 + 0.80 * (-0.3) + .525 * (-0.2) + 0.11 * (-0.4) + 0.907 * 0.5 + (-0.4) = -0.1433$
$I_8 = 0.894 * (-0.3) + 0.134 * (-0.5) + 0.80 * (-0.2) + 0.525 * 0.3 + 0.11 * 0.3 + 0.907 * (-0.3) + 0.2 = -0.2768$
$I_9 = 0.894 * 0.4 + 0.134 * 0.2 + 0.80 * 0.1 + 0.525 * (-0.1) + 0.11 * (-0.2) + 0.907 * (-0.5) + 0.5 = 0.4364$
$I_{10} = 0.4642*0.2+0.4013*0.4+0.2689*(-0.4)+0.3 = 0.4554$
Output $O_7 = \frac{1}{1+e^{0.1433}} = 0.464236$
$O_8 = \frac{1}{1+e^{0.2768}} = 0.431238$; $O_9 = \frac{1}{1+e^{0.4364}} = 0.646359$

Table 1: The Errors for Hidden/Output Units

Unit j	Err_j
10	$0.6119x(1-0.6119)x(1-0.6119) = 0.0921$
9	$0.6074x(1-0.6074)x(0.0921x(-0.4)) = -0.0038$
8	$0.4312x(1-0.4312)x(0.0921x0.4) = 0.0090$
7	$0.4642x(1-0.4642)x(0.0921x0.2) = 0.0045$

$O_{10} = \frac{1}{1+e^{0.4554}} = 0.6119$
The error of each unit is computed and propagated backward by updating the weights and biases to reflect the error of the network's prediction such that for a unit j in the output layer with a target output T_j and actual output O_j, the error Err_j is computed by $Err_j = O_j(1-O_j)(T_j - O_j)$.
The computed error of a hidden layer unit j is the weighted sum of the errors of the units connected to unit j in the next layer, such that given a hidden layer unit j, where w_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, and Err_k is the error of unit k, then error of the hidden layer unit j Err_j is computed with the following formula. $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$.
Table 1 shows the computed errors for hidden layer units 7, 8 and 9 as well as the output unit 10 of the example MLP network of Figure 2. The weights and biases are updated after the presentation of each tuple for case updating although these increments alternatively could be accumulated and updated after all tuples in the training set are presented for epoch updating. The algorithm terminates when all weight increases Δw_{ij} in the previous epoch or collection of training tuples were smaller than a specified threshold or when the algorithm has processed a specified number of epochs (or run long enough through a specified number of iterations). In practice, hundreds of thousands of epochs may be needed for the weights to converge. With the learning rate l ($0 \leq l \leq 1$, too low a learning rate causes slower learning while too high a learning rate causes oscillations of inadequate solutions), changes in weights (δw_{ij}) and biases ($\delta\theta_j$), the weight w_{ij} and bias θ_j for hidden layer unit j that is connected to an input lower layer unit i, are updated to reflect the propagated error using the following equations.
$\delta w_{ij} = (l) Err_j O_i$; $w_{ij} = w_{ij} + \delta w_{ij}$
$\delta\theta_j = (l) Err_j$; $\theta_j = \theta_j + \delta\theta_j$
An example updating of the weights w_{47} and w_{58} as well as biases for the units 7, 8, 9 and 10 are given below. $w_{47} = w_{ij} + (l) Err_j O_i = -0.2 + (0.9)(0.0045)(0.525) = -0.1978$
$w_{58} = w_{ij} + (l) Err_j O_i = 0.3 + (0.9)(0.0090)(0.11) = 0.3008$
$\theta_{10} = \theta_{10} + (l) Err_j = 0.3 + (0.9)(0.6119) = 0.8507$
$\theta_9 = \theta_9 + (l) Err_j = 0.5 + (0.9)(-0.6074) = -0.0466$
$\theta_8 = \theta_8 + (l) Err_j = 0.2 + (0.9)(0.4312) = 0.5880$
$\theta_7 = \theta_7 + (l) Err_j = -0.4 + (0.9)(0.4642) = 0.0177$
The algorithm is successfully finished, if the sum of the squared output errors for each pattern is zero (perfect) or approximately zero. The equation for calculating the root mean square error for a series of n values of z (e.g., output error values) is as follows:
$z_{rms} = \surd(z1^2 + z2^2 + \ldots + z^n)/n)$. For example, with the 4 output units of our experimental MLP network, the root means square error after the first round of propagation using the Err_7 to Err_{10} computed above is: $Err_{rms} = \surd((0.09121^2 + -0.003^2 + 0.009^2 + 0.0045^2)/4)$. This is equal to $\surd 0.00215 = 0.046$. After a repeated training and learning, we got the root mean square error for training data, $Err_{rms} = 0.0058$, which is matched very closely with the desired root mean square (RMS) error of 0.0. An anomaly score is assigned to each packet after calculating the difference between its output error and a set threshold. If the anomaly score is positive, then the relevant wireless packet is flagged as "Anomalous" and an attack number is assigned to it for future comparison with incoming packets. If the anomaly score is zero or close to zero, the packet is flagged as "Unknown" packet and is sent back into the training process. Finally, if the anomaly score is negative, then the packet is flagged as "Normal". Then, these anomalous data packets are stored in the intrusion database and an attack number is assigned to each identified anomalous packet. The system sends back the "Unknown" packets into the training process for re-evaluation. This re-evaluating process dynamically enhances the intrusion database of our proposed system.

3. CONCLUSIONS AND FUTURE WORK

This paper developed a Neural Network based wireless intrusion detection system for misuse detection based on learning and building quality training data for better detection of anomalous packets. The paper also shows the advantages of using hardware sensor to monitor real-time traffic in order to improve intrusion response time. Experiments comparing NeuDetect with 3 other wireless intrusion systems, Snort Wireless, Wifi-Miner [5] and WIDs [1], show that proposed system was able to detect more known intrusions than the other systems. In detection of known intrusion, our system has a better performance with high correct detection rate and a low false alarm rate.

4. ACKNOWLEDGMENTS

This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant.

5. REFERENCES

[1] C. I. Ezeife, M. Ejelike, and A. K. Aggarwal. Wids: A sensor-based online mining wireless intrusion detection system. In *Proceedings of the 12th ACM International Database Engineering & Applications Symposium (IDEAS08)*, pages 255–262. ACM, Sept 2008.

[2] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, New York, 2000.

[3] Y. Liu, D. Tian, and B. Li. A wireless intrusion detection method based on dynamic growing neural network. In *IEEE First International Multi-Symposiums on Computer and Computational Science - IMSCCSs*, pages 611 – 615, June 2006.

[4] NetworkChemistry. Network chemistry wireless security business. http://www.networkchemistry.com, 2007.

[5] A. Rahman, C. I. Ezeife, and A. K. Aggarwal. Wifi miner: An online apriori-infrequent based wireless intrusion detection system. In *Proceedings of the 2nd ACM SIGKDD '08 International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD, 2008)*, pages 77 – 89. ACM, August 2008.