ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2014, T. 43, Nr. 4

Network Intrusion Detection Using Self-Recurrent Wavelet Neural Network with Multidimensional Radial Wavelons

V. Alarcon-Aquino, J. M. Ramirez-Cortes[1], P. Gomez-Gil[1], O. Starostenko, Y. Garcia-Gonzalez

Department of Computing, Electronics, and Mechatronics
Communication and Signal Processing Group
Universidad de las Americas Puebla
72810 Cholula, Puebla, Mexico
e-mail: vicente.alarcon@udlap.mx

[1] Department of Electronics and Computer Science
National Institute of Astrophysycs, Optics and Electronics
Tonantzintla, Puebla, Mexico

crossref http://dx.doi.org/10.5755/j01.itc.43.4.4626

Abstract. In this paper we report a novel application-based model as a suitable alternative for the classification and identification of attacks on a computer network, and thus guarantee its safety from HTTP protocol-based malicious commands. The proposed model is built on a self-recurrent neural network architecture based on wavelets with multidimensional radial wavelons, and is therefore suited to work online by analyzing non-linear patterns in real time to self-adjust to changes in its input environment. Six different neural network based systems have been modeled and simulated for comparison purposes in terms of overall performance, namely, a feed-forward neural network, an Elman network, a fully connected recurrent neural network, a recurrent neural network based on wavelets, a self-recurrent wavelet network and the proposed self-recurrent wavelet network with multidimensional radial wavelons. Within the models studied, this paper presents two recurrent architectures which use wavelet functions in their functionality in very distinct ways. The results confirm that recurrent architectures using wavelets obtain superior performance than their peers, in terms not only of the identification and classification of attacks, but also the speed of convergence.

Keywords: Self-Recurrent Wavelet Neural Networks; Multidimensional Radial Wavelons; Intrusion Detection Systems.

1. Introduction

It is a matter of common knowledge that network security poses a constantly evolving and increasingly complex task due to the sheer size of the distribution and array of computer network interconnections [1]. Certain applications, such as online banking sites, are particularly at risk due to the fact that a breach in security would be catastrophic. As a result, several approaches (see e.g. [1–5]) have already been developed as alternative solutions to the problem of Internet security, focusing for the most part in detecting attacks and thus enabling pertinent corrective or preventive measures to be taken.

The classification carried out by intrusion detection systems may take into account the information source and the analysis technique. The information source uses the origin of the data that is analysed in order to determine whether an attack has been carried out or not, whereas the analysis technique is the method by which the data obtained by the information source will be analyzed. Intruder Detection Systems (IDSs) may obtain the information for the analysis from different sources. When an IDS only considers the data obtained from isolated terminals, it is classified as host-based. They will carry out statistical analysis of data within the host, carrying out access control routines and evidence collection of suspected attacks as part of their normal functions [1]. Due to their thorough approach, host-based systems are widely used in intranet environments where the threat of a security breach comes from inside the network, not from an external intruder. In contrast, when an IDS evaluates the flow of information that travels through a network, the IDS is classified as network-based.

The analysis techniques used in IDS are classified as the misuse intrusion detection and anomaly intrusion detection. The misuse intrusion detection utilizes a database containing attack signatures, where each signature corresponds to a specific attack and describes its characteristics. This technique must maintain the updating of the database that contains the attack signatures as it can only recognize attacks whose profile coincides with a signature on the database (see e.g., [6]). The main disadvantage of this analysis technique is that it can only detect attacks that are already known and whose pattern has been previously described. One of the advantages of IDSs based on misuse technique is that they obtain results with a low rate of false positives. On the other hand, anomaly intrusion detection analyzes the state and behavior of the system to categorize normal behavior and anomalies. The main advantages of the technique based on anomalies are that it is not necessary to establish a database of the characteristics of each attack and that it can detect attacks that have not been seen before (see e.g., [7, 8]). However, the main disadvantage of a technique using this focus is the volatility of the network's normal operating state, which can lead to a considerable rate of false positives [6].

It is the very dynamic and ever-changing nature of computer network attacks that make an approach based on neural networks an efficient course of action. Since neural networks are essentially an array of massively interconnected parallel processing elements they excel in pattern recognition, classification and parallel computation tasks [4]. Neural networks possess characteristics that make them an attractive tool for intrusion detection. Intrusion detection systems carry out a prediction mechanism based on known and unknown patterns given that the neural network models permit the definition of an interconnection scheme which, after being trained, can detect abnormal behavior. Naturally this leads to the investigation of different types of neural network architecture and learning algorithms in IDS. Within the analysis technique based on anomalies, attempts are made to detect anomalous behavior in normal activity. Several approaches have been used to treat intrusion detection systems based on anomalies: statistical, generation of predictive patterns, and artificial neural networks [1, 9].

Artificial neural networks allow computers to learn and adapt to the different tasks with which they are presented, and are inspired by the manner in which the human brain functions. Interconnected neurons manifest a response depending on the input stimulus. The stimulus that the neurons receive on input is translated into a decision as an output of the neural network [4, 10]. To carry out decisions, the neural network requires a phase of iterative training, in which data samples are applied and weights are adjusted until the resulting factor is found to be close to the desired result. Owing to their characteristics of learning and generalization, a neural network can contribute to the implementation of an intrusion detection system that allows the detection of a variety of attacks – both known and never seen before [11].

The incorporation of neural networks into intrusion detection systems has experimented with different architectures and learning algorithms. Initial studies can be found, such as [5], which present some of the advantages and disadvantages of the application of neural networks in IDS. In [11], the study implements an intrusion detection system called NNID (Neural Network Intrusion Detection), which uses a collection of commands executed by the user as the object of the anomalies, and whose objective is that the neural network identifies the variations in the patterns of use. In [9], a prototype named I-IDS (Intelligent Intrusion Detection System) is developed, which monitors the flow of packets and classifies network events using a multilayer perceptron (MLP) neural network trained with the back-propagation (BP) algorithm, which takes the hyperbolic tangent as its transfer function. Comparative studies of different neural network architectures applied to intrusion detection systems can be found in [12, 13]. These studies include MLP/BP, Radial Basis Function, Self-Organizing Maps, and Layered Framework with Neural Networks.

The different studies described above analyze performance indices in terms of false positives and false negatives. Common problems are identifiable within the results that reduce the performance of the neural networks applied to intrusion detection systems: low convergence speed, the local minima, the difficulty of determining the number of hidden layers, the quantity of units in the layers, and the quantity of samples necessary during the training. Simulated annealing and genetic algorithms have been used to overcome such difficulties as the local minima, and to accelerate the training process through the Levenberg-Marquardt algorithm and the algorithm used by the conjugate gradient method.

It is evident that a neural network based IDS must have the capacity to work online and update its parameters in real time to adapt to the flow of data over a period of time in a real network. This is one of the reasons why, out of all the possible neural network architectures, recurrent neural networks (RNN) hold the best performance parameters in classification, identification and convergence speed [4, 10, 14]. In the following sections, six different neural network based systems for computer IDS are proposed and analyzed, namely, a feed-forward neural network system (FFNN), an Elman neural network system, a fully recurrent neural network (FRNN) system, a self-recurrent wavelet neural network (SRWNN) system, a self-recurrent wavelet neural network with multidimensional radial wavelons (SRWNN-MRW) and a wavelet recurrent neural network (WRNN) system. Through the comparison of performance percentages in classification and identification, it will be shown

that the approach that has been followed throughout this paper justifies the use of wavelets and neural networks as a powerful base for an IDS which works directly on the causes of an attack through HTTP commands that may be malicious in nature, thus protecting the network data flow from unauthorized command insertion.

The main advantages of the proposed IDS approach based on the SRWNN-MRW respect to previous neural network-based methods are the following. The first one is the lower complexity of the overall detection model which directly depends on the reduced complexity of the proposed recurrent neural network. Note that, as it will be discussed in the performance evaluation section of this paper, SRWNN-MRW requires a significantly smaller number of neurons to obtain a similar performance than previously proposed neural-networks architectures. The second advantage is an improvement in the detection speed of the system which is supported by the fact that SRWNN-MRW requires a reduced number of iterations to reach convergence. This allows faster online learning and, as a consequence, faster detection and identification of HTTP attacks is possible.

An additional aspect that is important to highlight here is that SRWNN-MRW is a new neural network topology which is different from previous WNN approaches in the way multidimensional data are treated. In order to apply traditional WNN into the context of multidimensional data multidimensional wavelets are usually employed. These multidimensional wavelets are usually constructed using tensor products of one-dimensional wavelets. However, as it has been discussed in [15], one of the key issues for WNN is that the complexity of each wavelet considerably increases with the dimension. In contrast, in the proposed approach, the use of the so called MRW units which only employs one-dimensional wavelets reduces the complexity of the network. The remainder of this paper is organized as follows. Section 2 discusses the HTTP protocol. In Section 3 we propose the IDS to detect and classify attacks in high-level network protocols through the use of recurrent neural networks and wavelets. In Section 4, we focus on the simulation results of each system. In Section 5 conclusions and future work are presented.

2. Description of HTTP Protocol

The Spanish company s21sec [16] states that the Hypertext Transfer Protocol (HTTP) protocol has become the most widely used protocol in the field of computer communications. This makes it an obvious target for computer network intruders, as it provides a way to access restricted networks through the use of modified commands which may go undetected if the traffic flow between end systems is not monitored. Any IDS that attempts to successfully identify and classify HTTP based attacks must then be provided with a comprehensive database in order for the system to have a starting point from which to begin deriving its own normal and intrusive data pattern comparison sequences [1]. The database must be representative of both, the main types of attacks and the main types of normal commands that can be present in a network. In this study, five main categories of possible data flow have been considered [3]:

Normal: It includes the normal behavior of a system command without any attack involved.

Command Injection: It includes commands (shell codes written in machine language) executed directly on the system.

SQL Database Attack: It includes commands executed on SQL databases.

XSS (Cross Site Scripting): It includes commands executed via HTML, Java or JavaScript. This technique involves scripts that are designed to extract information from the user and pass it to the attacker. XSS is a type of attack that has as target sites those that dynamically generate web pages that display user entries that have not been properly validated. An attacker that uses XSS can compromise sensitive information, manipulate cookies or run malicious code on the client.

Path Modification: It considers the path manipulation of a file or directory to provide privileges to the attacker.

The IDS reported in this paper has the ability to correctly identify an attack or normal command, and then classify it under one of the five possible labels that have been considered.

3. Proposed Approach: SRWNN with Multidimensional Radial Wavelons

The architecture proposed in this work is composed through the use of processing nodes known as multidimensional radial wavelons (MRW), which were first proposed in [15, 17] in a forward propagation power architecture designed for the approximation of multivariable functions. Fig. 1 shows the architecture of a MRW unit. Each MRW unit is composed of two modules, the first of which has the radial function denoted by the letter R, and the second of which has the one-dimensional wavelet function denoted by ψ. The purpose of this division is to process the inputs vector so that they can be used for a one-dimensional wavelet function.

Recall that wavelet transforms involve representing a general function in terms of simple, fixed building blocks on different scales and at different positions. These building blocks are generated from a single fixed function called a mother wavelet ψ by translation and dilation operations. The wavelet transform can be considered a family of related functions [18]:

$$\psi_{d,t}(s) = \frac{1}{\sqrt{|d|}}\psi\left(\frac{s-t}{d}\right) \quad (1)$$

where $d \in \mathbb{R}^+$, $t \in \mathbb{R}$, $d \neq 0$, and $\psi(\cdot)$ satisfies the admissibility condition. For discrete wavelets, the scale (or dilation) and translation parameters in (1) are chosen in such a way that at level j the wavelet $d_0^j \psi(d_0^{-j} s)$ is d_0^j times the width of $\psi(\cdot)$. That is, the scale parameter $\{d = d_0^j : j \in Z\}$ and the translation parameter $\{t = nt_0 d_0^j : j, \in Z\}$ This family of wavelets is thus given by

$$\psi_{j,n}(s) = \mathrm{d}_0^{-\frac{1}{2}} \psi\left(d_0^{-j} s - nt_0\right). \quad (2)$$

The mother wavelet function $\psi(s)$, scaling d_0 and translation t_0 parameters are specifically chosen so that $\psi_{j,n}(s)$ constitute orthonormal bases for $L^2(\mathbb{R})$ [18]. Fig. 2 shows the architecture of the proposed SRWNN – MRW model. The proposed architecture is composed of three layers; the first is the input layer, the second is the multidimensional radial wavelon layer, and finally the output layer, composed of linear combiners. The external inputs are represented by x_k, $k = 1 \ldots N_i$, N_i being the number of inputs defined for the architecture. The outputs of the R units are denoted by a_j, $j = 1 \ldots N_{MRW}$, N_{MRW} being the number of multidimensional radial wavelon units. These outputs represent the processing of the input lines. Within the multi-dimensional radial layer, the wavelet units possess an auto-connection that permits the introduction of the term memory $\psi_j(n-1)$ at time n, which is responsible for the saving the information regarding past states of the model. This recurring connection is seen as affected by means of the factor θ_j, which represents the synaptic weight of this link. That is to say, the input at the wavelet units is formed by the output from the units R and the term $\psi_j(n-1)$ multiplied by the factor θ_j.

The direct connections from the input layer to the output combiners are affected by factors a_{ik}, $i = 1 \ldots N_s$, N_s being the number of output units. The synaptic weights between the output units and the wavelet units are denoted by w_{ij} .The input layer receives the external inputs and transmits them directly to the multidimensional radial layer. It is possible to denote the vector $\boldsymbol{x}_k$ of external inputs by means of

$$\boldsymbol{x}_k = [x_1 \ldots x_{N_i}] \quad (3)$$

where N_i represents the number of external inputs.

The MRW units are found within multidimensional radial wavelon layer. Each unit manages its own translation t_j and dilation parameters d_j for each of the inputs in $\boldsymbol{x}_k$. It is possible to represent these parameters for the MRW unit in a vector form, as in the following:

$$\mathbf{d}_j = [d_{j,1} \ldots d_{j,N_i}] \quad (4)$$

$$\mathbf{t}_j = [t_{j,1} \ldots t_{j,N_i}] \quad (5)$$

$$j = 1 \ldots N_{MRW}$$

where N_{MRW} represents the number of multidimensional radial wavelon units found in this layer. The principal task of the units R within this layer is to process the multiple inputs in order to produce a scalar value, which serves in the construction of the input to the wavelet unit.

To develop the processing of the units R, it is necessary to define:

$$A(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j) = diag(\mathbf{d}_j)(\mathbf{x}_k - \mathbf{t}_j)^T \quad (6)$$

where T denotes the transposed operation of the difference between the inputs vector $\mathbf{x}_k$ and the translation vector $\mathbf{t}_j$, $diag(\boldsymbol{d}_j)$ represents a diagonal matrix constructed from the dilation vector $\mathbf{d}_j$, as can be seen as follows:

$$diag(\mathbf{d}_j) = \begin{pmatrix} d_{j,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{j,N_i} \end{pmatrix}. \quad (7)$$

From $A(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j)$ the output of each unit R is constructed in the following manner:

$$R(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j) = \left[A(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j)^T A(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j)\right]^{\frac{1}{2}} \quad (8)$$

$$a_j = R(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j). \quad (9)$$

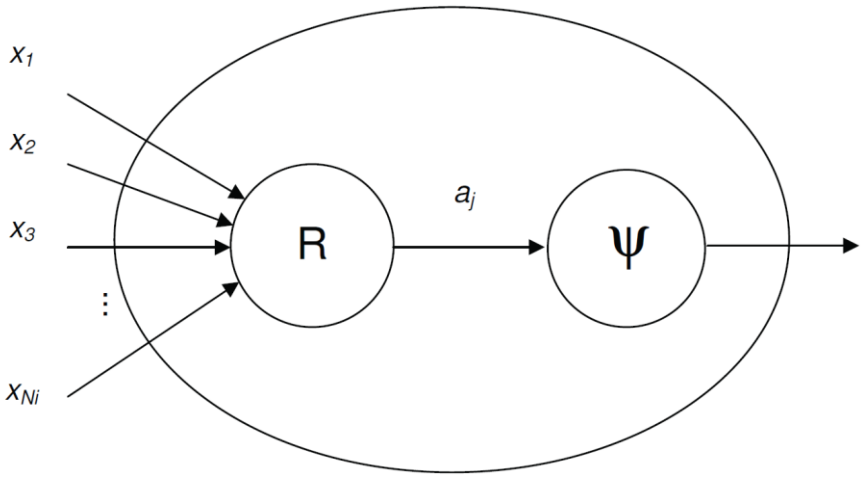

Figure 1. Multidimensional Radial Wavelon Unit

Having defined the outputs of the units R (represented by scalar values), a recursive connection between the wavelet units is introduced. This connection is affected by the synaptic weight θ_j in the following manner:

$$\alpha_j = a_j + \psi_j(n-1)\theta_j \quad (10)$$

where α_j represents the input of the wavelet unit $\psi(n)$. Finally the output of the wavelet unit is given by

$$\psi(\alpha_j) = \psi(a_j + \psi_j(n-1)\theta_j) \quad (11)$$

The output layer is comprised of linear combiners, which represent the distinct outputs of the model. Each output unit has a weight associated with each unit of the previous layer (w_{ij}) and with each input unit (a_{ik}). The synaptic weights are defined respectively by

$$w_{ij}, i = 1 \dots N_s, j = 1 \dots N_i \quad (12)$$

$$a_{ik}, i = 1 \dots N_s, k = 1 \dots N_i \quad (13)$$

where N_s represents the number of defined outputs in the architecture. The value of each output unit is given by

$$y^i(n) = \sum_{j=1}^{N_{MRW}} w_{ij}\, \psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\, x_k. \quad (14)$$

The training method for SRWNN – MRW is based on the gradient-descent and seeks to minimize the following quadratic cost function:

$$J_I = \frac{1}{2}[y^d(n) - y^i(n)]^2 \quad (15)$$

where $y^d(n)$ is the target value for that output and $y^i(n)$ is the resulting value of the system at time n. The tuning parameters that belongs to the vector $W = [a_{ik} t_{jk} d_{jk} \theta_j w_{ij}]$ of the SRWNN-MRW are continually adjusted to minimize the error obtained after a certain number of training cycles. The delta rule determines the amount of update for the tuning parameters based on the gradient direction along with its learning rate η_I, as follows:

$$W(n+1) = W(n) + \Delta W(n)$$
$$= -\eta_I \left(\frac{\partial E}{\partial W(n)}\right). \quad (16)$$

The partial derivative of the cost function with respect to the tuning parameters is as follows:

$$\frac{\partial E}{\partial W(n)} = \sum_{k\in 0} e_k \frac{\partial y^i(n)}{\partial W(n)} \quad (17)$$

We need to calculate the value of $\frac{\partial y^i(n)}{\partial W(n)}$ to each one of the tuning parameters that belongs to the vector $W = [a_{ik} t_{jk} d_{jk} \theta_j w_{ij}]$.

a) Weight between the input layer and output layer

The partial derivative of $\frac{\partial y^i(n)}{\partial a_{ik}(n)}$ is computed by using the output of the system as follows:

$$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\, x_k}{\partial a_{ik}(n)}. \quad (18)$$

The weights between the final two layers and the output of the multidimensional units do not depend on

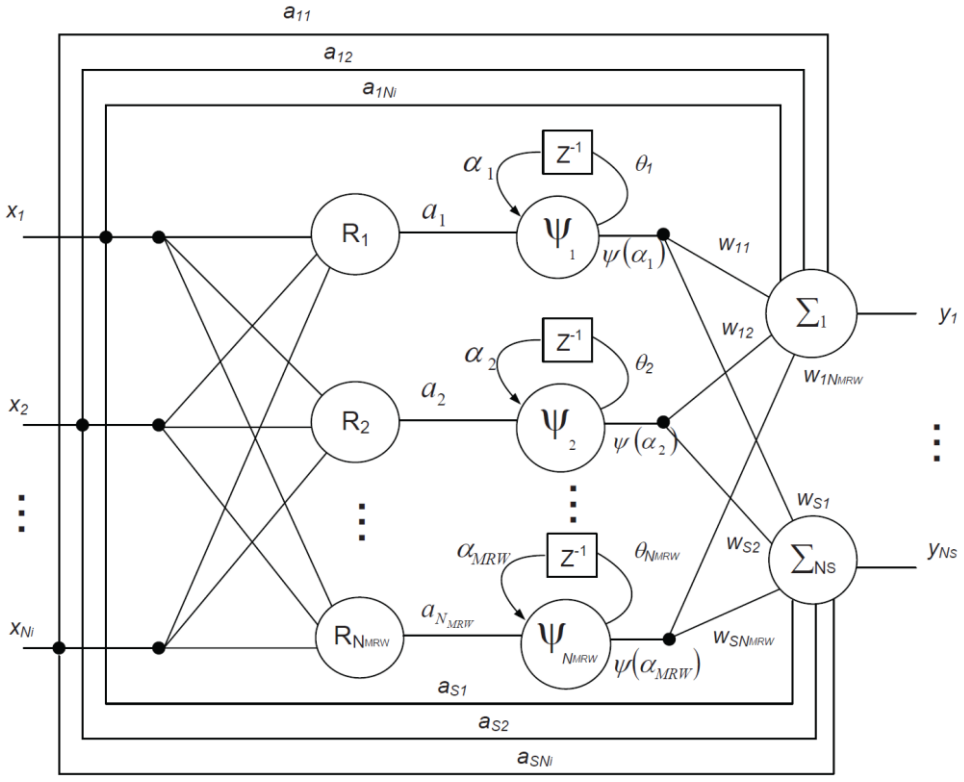

Figure 2. Proposed SRWNN-MRW architecture

$a_{ik}(n)$. Therefore the previous equation is converted into

$$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = \frac{\partial \sum_{k=1}^{N_i} a_{ik}\, x_k}{\partial a_{ik}(n)} = x_k. \quad (19)$$

Equation (19) is the partial derivative of the output with respect to $a_{ik}(n)$ in the SRWNN – MRW model.

b) The translation parameters

For the calculation of $\frac{\partial y^i(n)}{\partial t_{jk}(n)}$, it is necessary to define

$$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\, \psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\ x_k}{\partial t_{jk}(n)}$$

From this equation, the second term in the addition does not depend on the parameters of translation of the units of the wavelet layer, so that it is converted into

$$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = \frac{\partial \sum_{i=j}^{N_{MRW}} w_{ij}\, \psi(\alpha_j)}{\partial t_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial t_{jk}(n)}$$

By the chain rule, we obtain

$$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial t_{jk}(n)}$$

Developing the third term of the product we have,

$$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \frac{\partial \left\{ d_{j1}(x_1 - t_{j1})^2 \ldots d_{jk}(x_k - t_{jk})^2 \right\}^{\frac{1}{2}}}{\partial t_{jk}(n)}$$

which gives

$$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \left[\frac{1}{2a_j} (-2d_{jk}^2 x_k + 2d_{jk}^2 t_{jk}) \right] \quad (20)$$

with $a_j = R(\mathbf{x}_k, \mathbf{d}_j, \mathbf{t}_j)$.

Equation (20) is the partial derivative of the output with respect to $t_{jk}(n)$ in the SRWNN-MRW model.

c) The dilation parameters

Likewise it can be established

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij} \psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\, x_k}{\partial d_{jk}(n)}.$$

From this equation, the second term in the addition does not depend on the parameters of dilation of the units of the wavelet layer, so that it is converted into

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\, \psi(\alpha_j)}{\partial d_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial d_{jk}(n)}$$

By the chain rule, it follows that

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial d_{jk}(n)}.$$

Developing the third term of the product we have,

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \frac{\partial \left\{ d_{j1}(x_1 - t_{j1})^2 \ldots d_{jk}(x_k - t_{jk})^2 \right\}^{1/2}}{\partial d_{jk}(n)}.$$

Then, it follows that

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \left[\frac{1}{2a_j} \left(2d_{jk} x_k^2 - 4d_{jk} x_k t_{jk} + 2d_{jk} t_{jk}^2 \right) \right]. \quad (21)$$

Equation (21) is the partial derivative of the output with respect to $d_{jk}(n)$ in the SRWNN – MRW model.

d) Weights of the self-recursive connections

To begin, we define the value of

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\, \psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\, x_k}{\partial \theta_j(n)}$$

The second term in the addition does not depend on the factor $\theta_j(n)$, so that it is converted into

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = \frac{\partial \sum_{i=1}^{N_{MRW}} w_{ij} \psi(\alpha_j)}{\partial \theta_j(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \theta_j(n)}.$$

By the chain rule, it follows that

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial \theta_j(n)}.$$

This yields,

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij} \frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\, \psi_j(n-1). \quad (22)$$

Equation (22) is the partial derivative of the output with respect to $\theta_j(n)$ in the SRWNN-MRW model.

e) Weights of the connections between the product layer and the output layer

To begin, we define the value of

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij} \psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik} x_k}{\partial w_{ij}(n)}.$$

Now, it is possible to calculate the partial derivative as follows:

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\, \psi(\alpha_j)}{\partial w_{ij}(n)}$$

This yields,

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \psi(\alpha_j). \quad (23)$$

Equation (23) is the partial derivative of the output with respect to $w_{ij}(n)$ in the SRWNN-MRW model. To sum up, the adjustments to each of the tuning parameters that belong to the vector W are shown in Table 1.

4. Performance Evaluation

All HTTP requirements that the intruder detection system will come across belong to one of the five categories under consideration. Ideally, the system

Table 1. Partial derivatives for the proposed SRWNN-MRW

Partial derivative of the output $\partial y^i(n)$ with respect to	**Proposed SRWNN-MRW Model**
$\partial a_{ik}(n)$	$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = x_k$
$\partial t_{jk}(n)$	$\frac{\partial y^i(n)}{\partial t_{jk}(n)} = w_{ij}\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\left[\frac{1}{2a_j}\left(-2d_{jk}^2 x_k + 2d_{jk}^2 t_{jk}\right)\right]$
$\partial d_{jk}(n)$	$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij}\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\left[\frac{1}{2a_j}\left(2d_{jk}x_k^2 - 4d_{jk}x_k t_{jk} + 2d_{jk}t_{jk}^2\right)\right]$
$\partial \theta_j(n)$	$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij}\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\psi_j(n-1)$
$\partial w_{ij}(n)$	$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \psi(\alpha_j)$

will classify as well as detect all incoming input vectors correctly, thus allowing successful subsequent actions in response to an eventual attack. For training and testing purposes, the dataset to be utilized has been divided in such a way as to count with 488 requirements representing abnormal commands (Path, Injection, XSS and SQL) and 285 representing normal commands [2, 3]. The IDS is divided in two main sections: data preprocessing and neural network training and testing.

4.1. Data Preprocessing

The preprocessing procedure is described using an example of a typical HTTP requirement given by //name.exe?param1=..\..\file. Note that this string consists of filenames, parameters and alphanumeric strings which normally change from system to system. Since the file extensions and special characters are the most relevant parts of this string, every alphanumeric string is replaced with the special character '@' [2]. Hence, the example requirement takes the form: //@.exe?@=..\..\@, where only the significant part of the requirement is kept.

The next step involves converting these characters to their corresponding ASCII decimal value. Since neural network needs a fixed-length input vector and http requirements do not have a fixed length, the next step is fixing the requirement's length already formatted to decimal. For this purpose, a sliding window approach is employed, whose main idea is converting a variable length vector in several fixed length vectors, where the length is determined by a constant defined by the nature of the problem. The sliding window approach considers a decimal vector with six elements. Now suppose that a neural network requires a fixed input length M equal to three. Then, the sliding window approach delivers (N-M+1) vectors of fixed length M, where N is the length of the original vector. Note that in order to improve network performance, these vectors are then converted into their binary representation and then arranged into a binary matrix. Finally this matrix is expressed as a single vector of length $M \times 8$, where M is the fixed length defined by the sliding window [2].

4.2. Performance Metrics

To carry out the evaluation of the intrusion detection system we must establish indicators that allow to know the percentage of detected attacks, or intrusive behaviors, and the percentage of false alarms. Likewise that commercial IDS, the implementation can be affected by varying the alert threshold. This means that we must find a balance between detection rate and false alarm rate. In summary, we may adjust the parameters for each neural network topology to detect more attacks but this would also cause an increase in the rate of false alarms. The evaluation of the different models is mainly concentrated in two indicators. The first one is the false positive (FP) which indicates the number of false alarms that corresponds to patterns that are labeled as harmful when the IDS is actually dealing with harmless elements. The second indicator is the false negative (FN) which denotes dangerous or intrusive behaviors similar to normal events that are not detected by the IDS features. In addition to FP and FN, we can also define indicators of true positives (TP) and true negatives (TN). The former describes the correct classification of harmful patterns, while the latter denotes the correct classification of normal traffic. Besides the calculation of the above indicators is useful to define two additional measures. The first relates to the level of specificity or true negative rate (TNR) of the assessed model, that is, the percentage of patterns of network traffic that are not intrusive. The second measure relates to the sensitivity or true positive rate (TPR) and describes the detection rate of intrusive behaviors performed correctly. These performance metrics are computed as follows [19]:

$$Specificity(TNR) = \frac{TN}{TN+FP} \quad (24)$$

$$Sensitivity(TPR) = \frac{TP}{TP+FN}. \quad (25)$$

Table 2. Percentage of detection for several neural network architectures in each category

Network Type	Normal	Injection	Path	SQL	XSS
FFNN	97.10%	3.44%	95.89%	0%	53.96%
ELMAN	96.37%	10.34%	92.82%	42.85%	87.30%
FRNN	92.75%	58.62%	99.48%	28.57%	41.26%
WRNN	89.32%	60.40%	99.50%	36.24%	48.60%
SRWNN	92.75%	33.33%	99.48%	57.14%	65.07%
Proposed SRWNN-MRW	94.56%	33.33%	98.46%	57.14%	68.25%

The rate of false positives (FPR) and the rate of false negatives (FNR) are computed as follows [19]:

$$FPR = \frac{FP}{FP+TN} = 1 - Specificity \quad (26)$$

$$FNR = \frac{FN}{FN+TP} = 1 - Sensitivity. \quad (27)$$

4.3. Neural Network Architectures: Training and Testing

All neural networks described in this paper have an input vector length equal to 64 due to the size of the original sliding window procedure ($M \times 8$), which after the binary conversion requires such length in input patterns. As for the output layer, it has been decided that all networks are to bear five distinct neurons in this last output stage (see Section 2). Therefore, when a particular input sequence is identified, it may be classified into any of the categories through the onset of a value of 1 in the selected neuron, while all other neurons have a value of 0. Following these principles, the first network to be tested is a multilayer feedforward neural network (FFNN), with two hidden layers with 15 neurons each. The number of hidden layers and hidden neurons has been selected to allow for an optimal performance while keeping processing times within reasonable limits [4], [20]. This network is trained with gradient-descent with momentum and adaptive learning rule back-propagation algorithm, and all neurons are sigmoid to enable the network to output a one or a zero as this is a pattern classification problem. This neural network has been trained with 70% of the dataset and reached the estimated error value of 0.0839 upon completion of the training sequence. Table 2 shows the percentage of hits per category for each neural network model. It can be seen that for all assessed neural network architectures a lower detection rate in the category of injection and SQL is obtained, whereas a higher detection rate in the category of path is attained. If the category of normal traffic is observed, the table shows similar results for FRNN and self-recurrent neural networks using wavelets (SRWNN, SRWNNMRW), which translates into a similar percentage in the rate of false positives.

The second network to face training and analysis was a simple recurrent network, also known as Elman network. Following the principle previously stated for input and output layers, this network was nevertheless designed to bear 30 neurons in its first hidden layer and 30 neurons in its second layer. The reason for this increment in neurons, as has been explained in [20], is that feedback loops in the network architecture require a significantly greater number of processing nodes to accurately interrelate information from previous instants in time to the current expected output. Thus, following the number of input vectors and expected number of output neurons, the number of 30 neurons per layer was calculated [4]. The results from training are as expected; the network reached the minimum error value of 0.0537. It also did so in much less time, as measured by the number of epochs that were measured upon completion of its task (see Table 2 and Table 3).

The third network subject to evaluation was the fully connected recurrent neural network (FRNN). It bore the same number of neurons in its input and output layers, but the number of hidden layers was cut in half due to several criteria. Firstly, multiple hidden layers do give neural networks better time nonlinear phenomena processing qualities. However, an IDS is not a highly nonlinear phenomenon, as it is based merely on pattern classification, even if its inherent permutations are nonlinear [4], [11]. Secondly, training a RNN with the RTRL algorithm allows for much faster convergence upon the desired error minima, albeit at the cost of a significant increase in processing time. Following the calculation parameters in [9], the number of neurons per layer, 30, was assigned to the hidden layer, but the number thereof was set to one to reduce the processing time of the network as a whole. In small networks such as this, the total number of neurons is not a major determining factor in performance and speed [20]. However, better results than those of either of the previous networks were achieved by the FRNN even despite its having only one hidden layer. This architecture reached an error value of 0.0044 (see Table 2 and Table 3).

Another architecture modeled was a fully connected neural network based on wavelets (WRNN). The neural network was based on wavelets as part of the preprocessing of the training data as reported in [21]. The stationary discrete wavelet transform was used to decompose the training data in different levels of decomposition. Then we trained the recurrent neural network modeled above with

Table 3. Performance measures for several neural network architectures and the proposed approach

Network Type	FFNN 64×15×15×5	ELMAN 64×30×30×5	FRNN 64×30×5	WRNN 64×30×5	SRWNN 64×256×5	Proposed SRWNN – MRW 64×4×5
MSE	0.0839	0.0537	0.0044	0.0864	0.0589	0.0899
NMSE	7.33e-5	4.69e-5	3.92e-6	7.55e-5	5.14e-5	7.01e-5
Convergence speed (epochs)	493	160	81	100 each level	100	60
Classification	84.07%	86.14%	93.15%	95.67%	92.04%	92.19%
Identification	78.34%	81.84%	84.23%	92.64%	83.43%	84.23%
O(n)	N^2	N^2	N^4	N^4	N^2	N^2
FNR	26.13%	21.87%	6.53%	4.98%	8.52%	9.65%
FPR	2.89%	3.62%	7.24%	10.67%	7.24%	5.43%
Specificity	97.10%	96.37%	92.75%	89.32%	92.75%	94.56%
Sensitivity	73.86%	78.82%	93.46%	95.01%	91.47%	90.34%

random-initialized weights, with the last decomposition level for 100 epochs. After that, we trained for 100 epochs the same RNN, with the previous estimated weights, but now with the approach of the data reconstructed in one level. This process was repeated until the data was completely reconstructed. The algorithm used to train the RNN in all levels was the RTRL, with a modification proposed in [20] to reduce the complexity order to $O(n^3)$.

The fifth architecture presented is the Self-Recurrent Wavelet Neural Network (SRWNN).The topology of this network is found to be divided into four layers (see [22]): the input layer, the wavelon layer, the product layer, and finally the output layer. The external instances are accepted in the input layer and these inputs are then transmitted directly to the next layer. In the case of this study, 64 inputs were considered, which represent the length of each of the training samples for http requests. The first derivative of the Gaussian function $\psi(x) = -xexp(\frac{-1}{2}x^2)$ was chosen to be a mother wavelet. From this mother wavelet, we can calculate the output of each wavelet unit by making use of the parameters of translation and dilation assigned to each external stimulus, that is $\psi_{ij}(z_{ij}) = \psi((u_{ij} - t_{ij})/d_{ij})$, with $z_{ij} = \frac{u_{ij}-t_{ij}}{d_{ij}}$ where u_{ij} is the net input to the unit, t_{ij} is the translation factor and d_{ij} stands for dilation factor. The subscript ij describes the i th wavelet unit corresponding to the j th external input.

Each wavelon unit has a recursive connection to itself, which is affected by the synaptic weight, and in this way it introduces a memory term which saves information pertaining to previous states of the network. The wavelon unit outputs are concentrated in the product layer. Each one of the units in this layer can be considered multidimensional wavelon unit [15] and their output is the result of multiplying various one dimensional wavelets. Finally, the output layer is formed by various linear combinators whose inputs are formed by the outputs from the previous layer (the product layer) times a weight cost plus the inputs values times a weight factor. The training method for SRWNN is based on the gradient-descent with adaptive learning rates [23]. Further details on this architecture can be found in [22]. The best results obtained by the SRWNN architecture were with 4 wavelons per input, or 256 wavelons in total. During the training stage the SRWNN reached the error value of 0.0589 at epoch number 100 (see Table 2). The results obtained by this architecture in terms of false positive rate and false negative rate are close to those of FRNN but with a lower cost of processing time and storage capacity.

The last architecture presented is the proposed Self-Recurrent Wavelet Neural Network with Multidimensional Radial Wavelons (SRWNN – MRW) described in Section 3. The first derivative of the Gaussian function was chosen to be a mother wavelet used as activation function in the wavelons units. The number of wavelon units used in the architecture depends directly on the input domain established by the training samples. The initialization of the parameters of translation and dilation play a crucial part in ensuring that the wavelet functions extend over the entire input domain. The number of wavelons units in the SRWNN-MRW architecture was set to 4. The training method for SRWNN-MRW is based on the gradient-descent with adaptive learning rates [23]. During the training stage the SRWNNMRW reached the error value of 0.0899 at epoch number 60. The results obtained by this architecture established a superior in the rate of false positives compared with the FRNN and SRWNN models, having a rate of 5.43%.

All networks under analysis went through test periods in which 30% of the original dataset was used to assess their identification, classification and global

error performance parameters. At the output layer, a competence function was called upon. Identification and classification were used to evaluate network performance besides speed of convergence, cost, and number of neurons. Classification corresponds to the event where an input sequence has been accurately labeled as an attack (of any sort) or as a normal command sequence. This percentage takes into account only which of those inputs the network provides are correctly labeled in those two main categories, and does not take into consideration whether the particular command at an instant in time was correctly labeled according to its origin out of the four possible kinds of attack. Identification, on the other hand, concerns itself with the correct placing of any input pattern in its correct slot from out of the five. It should be evident that the expected percentages for identification are lower than those of classification, and in practice it was indeed so. In practice, too, it may have occurred that while classification was correctly assigned, identification itself failed, and therefore identification is in fact harder to attain by an IDS.

Table 3 summarizes all neural network performance parameters after testing and provides two final measures of network effectiveness: false positives and false negatives rates. True to their names, a false positive will occur when the network mistakenly classifies an HTTP command as an intrusion when it is not, and a false negative conversely involves the classification of an attack as normal network behavior. In overall terms, the WRNN has outperformed all other networks in all parameters, reaching a classification percentage of over 95% and thus clearly indicating its superiority when compared to other architectures for this particular application. In terms of convergence speed, a model based on a self-recurrent neural network SRWNN–MRW needed 60 training periods compared to the 81 periods required by the FRNN model, which leaves the Elman model in last place with 160. However, the computational cost required for the totally connected recurrent model is in the order of $O(n^4)$, when cataloguing the most costly in terms of computational resources. That is to say that taking into account the indices of classification and identification, the self-recurrent model using wavelets SRWNN – MRW, with a computational complexity of $O(n^2)$, offers an approximation close to the totally connected recurrent model at lesser computational cost.

The measurements in the rates of false positives and false negatives indicate that although the Elman model obtained a lower rate of emitted false positives, it also recorded the highest rate of false negatives of all the models. It is possible to analyze this behavior with the statistical indicators of specificity and sensitivity [19]. As stated previously, specificity is defined as the probability that an inoffensive input pattern is catalogued by the IDS as normal network traffic. Sensitivity is the probability that the IDS classify a damaging input pattern as an attack on the system. Once these two probabilities were identified it was clear that the Elman model possesses a high probability of specificity, around 96%, which explains the low rate of false positives. On the other hand, the Elman model possesses low sensitivity, 78%, which explains the high percentage in its false negative rate. In an ideal IDS, the specificity and sensitivity indicators should be high, although practical applications never obtain 100%, which does result in the emission of false positives and false negatives [19]. Taking into account the above, the totally

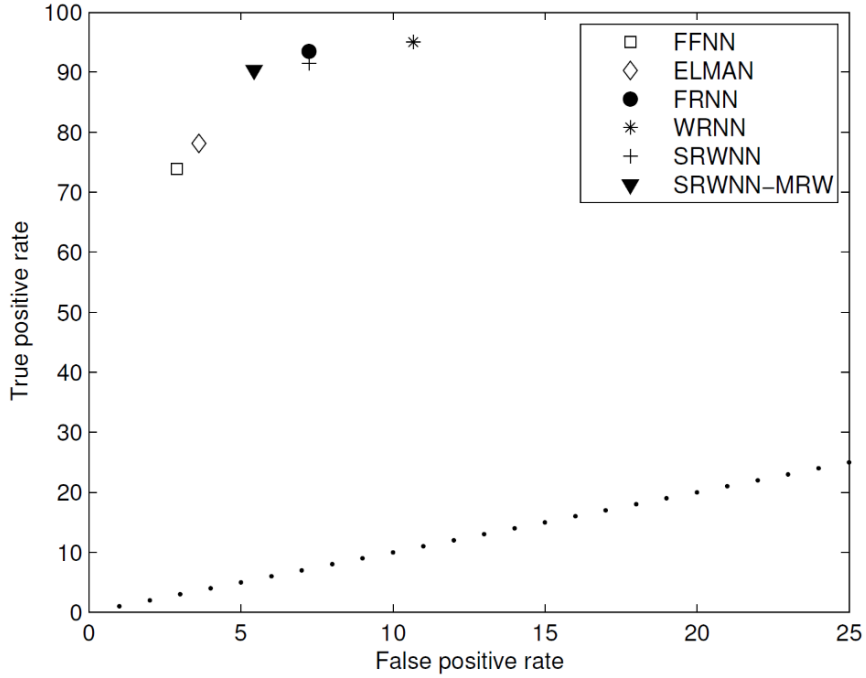

Figure 3. ROC Graph

connected recurrent model (FRNN) and the self-recurrent model using wavelets (SRWNN) achieve results with more equality between false negatives and false positives, which are 7.24% and 6.53% for the first, and 7.24% and 8.52% for the second model. To reiterate, the rates obtained by the proposed model (SRWNN-MRW) were 5.43% and 9.65%, establishing a superior in the rate of false positives than that of both architectures mentioned above.

The ROC (Receiver Operating Characteristics) graph in Fig. 3 shows that the application of neural networks to the models of intrusion detection performs in a significant way, due to the fact that the three models considered (FRNN, SRWNN, SRWNNMRW) are located in the upper triangular region. This permits them to establish their development at a distance from the random behavior found at the line $y = x$ [19]. With regard to cost, the graph shows that while the Elman model is found more in the West part of the graph, the remaining models considered are located more to the east, indicating greater emission levels of false positives. If the axis of the true positives is considered, it is clear that the totally connected recurrent model (FRNN) and the self-recurrent model using wavelets (WRNN, SRWNN and SRWNN–MRW) outperform the Elman and FFNN models considerably, establishing their advantages at the moment of detection of harmful patterns. The Elman model states that it is more conservative than the other models considered due to the fact that it carries out a positive classification of the patterns on the basis of solid evidence and as a result possesses a low rate of false positives, while at the same time having the disadvantage that it has a low rate of true positives. In contrast, the totally connected recurrent model (FRNN) and the self-recurrent models using wavelets (SRWNN and SRWNN-MRW) may be called liberal due to the fact that they carry out a positive classification on the basis of slighter evidence, resulting in a superior classification of attacks at the cost of obtaining a higher rate of false positives. As can be seen in the results outlined, the FRNN model and the SRWNN-MRW obtained the second best results in terms of the indicators of classification and identification, these being 93.15%, 84.23%, and 92.19%, 84.23%, respectively. These measurements identify these models as the most feasible candidates in the implementation of the intrusion detection system that may comply with the security levels required in a communication network. The self-recurrent model using wavelets (SRWNN-MRW) has been presented in this paper in order to observe its development as applied to intrusion detections systems.

5. Conclusions and Future Work

In this paper we have proposed a new model for the detection and classification of intrusions based on a self-recurrent wavelet neural network with multidimensional radial wavelons (SRWNN-MRW). This model is suited to work online by analyzing nonlinear patterns in real time to self-adjust to changes in its input environment. In terms of convergence speed, the model based on SRWNN–MRW needed 60 training periods compared to the 81 periods required by the FRNN model, which leaves the Elman model in last place with 160. However, the computational cost required for the totally FRNN model is in the order of $O(n^4)$, indicating the most costly in terms of computational resources. This means that taking into account the indexes of classification and identification, the SRWNN-MRW model, with a computational complexity of $O(n^2)$, offers an approximation close to the totally connected recurrent model at lesser computational cost. Future work will focus on improving the results obtained through the use of other recurrent schemes based on wavelets, for example, having a fully recurrent network with wavelons or using instead multidimensional radial wavelons.

Acknowledgments

This work was supported by National Council for Science and Technology of Mexico (CONACYT). The first author would like to thank Edgar S. Garcia-Treviño and the referees for their most helpful and constructive remarks.

References

[1] **A. Ghorbani, W. Lu, M. Tavallaee.** *Network Intrusion Detection and Prevention: Concepts and Techniques*. Advances in Information Security, Vol. 47. Springer, 2010.

[2] **V. Alarcon-Aquino, J. Mejia-Sanchez, R Rosas-Romero, J. Ramirez-Cruz.** *Detecting and Classifying Attacks in Computer Networks Using Feed-Forward and Elman Neural Networks*. EC2ND 2005. Springer, 2006, 187–196.

[3] **V. Alarcon-Aquino, C. A. Oropeza-Clavel, J. Rodriguez-Asomoza, O. Starostenko, R. Rosas-Romero.** *Intrusion Detection and Classification of Attacks in High-Level Network Protocols Using Recurrent Neural Networks*. Novel Algorithms and Techniques in Telecommunications and Networking. Springer, 2010, 129–134.

[4] **S. Haykin.** *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[5] **C. Bitter, D. A. Elizondo, T. Watson.** Application of artificial neural networks and related techniques to intrusion detection. *International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.

[6] **Z. Li, A. Das, J. Zhou, J. C. Patra.** Variable-length signatures for intrusion detection. *Computer Systems, Science and Engineering*, 2008, Vol. 23, No. 3, 183-191.

[7] **A. Tartakovsky, A. Polunchenko, G. Sokolov.** Efficient Computer Network Anomaly Detection by Changepoint Detection Methods. *IEEE Journal of Selected Topics in Signal Processing*, 2013, Vol. 7, No. 1, 4–11.

[8] **V. Alarcon-Aquino, J. A. Barria.** Anomaly detection in communication networks using wavelets. In: *IEE Proceedings-Communications*, IET, 2001, Vol. 148, No. 6, 355–362.

[9] **I. V. M. de Lima, J. A. Degaspari, J. B. M. Sobral.** Intrusion detection through artificial neural networks. *IEEE Network Operations and Management Symposium, NOMS*, 2008, 867–870.

[10] **P. Gomez-Gil, J. M. Ramirez-Cortes, S. E. P. Hernandez, V. Alarcon-Aquino.** A neural network scheme for long-term forecasting of chaotic time series. *Neural Processing Letters*. Springer, 2011, Vol. 33, No. 3, 215– 233.

[11] **J. Ryan, M.-J. Lin, R. Miikkulainen.** Intrusion detection with neural networks. *Advances in neural information processing systems*. Morgan Kaufmann Publishers, 2002, 943–949.

[12] **P. Kukielka, Z. Kotulski.** Analysis of different architectures of neural networks for application in intrusion detection systems. *International Multiconference on Computer Science and Information Technology*, *IMCSIT*, 2008, pp. 807–811.

[13] **N. Srivastav, R. Challa.** Novel intrusion detection system integrating layered framework with neural network. *IEEE 3rd International on Advance Computing Conference (IACC)*, 2013, pp. 682–689.

[14] **F.-J. Lin, Y.-S. Kung, S.-Y. Chen, Y.-H. Liu.** Recurrent wavelet-based Elman neural network control for multi-axis motion control stage using linear ultrasonic motors. *IET Electric Power Applications*, 2010, Vol. 4, No. 5, 314–332.

[15] **Q. Zhang.** *Wavelet network: The radial structure and an efficient initialization procedure*. Linkoping University, Sweden, 1993.

[16] **S21SEC**. *Expertos en seguridad digital*. 1999. url: http://www.s21sec.com.

[17] **Q. Zhang, A. Benveniste.** Wavelet networks. *IEEE Transactions on Neural Networks*, 1992, Vol. 3, No. 6, 889–898.

[18] **S. Mallat.** *A wavelet tour of signal processing: the sparse way*. Academic Press, 2008.

[19] **T. Fawcett.** *ROC graphs: Notes and practical considerations for researchers*. HP Laboratories, Palo Alto, Kluwer Academic Publishers, 2004.

[20] **M.-W. Mak, K.-W. Ku, Y.-L. Lu.** On the improvement of the real time recurrent learning algorithm for recurrent neural networks. *Neurocomputing*, 1999, Vol. 24, No. 1-3, 13–36.

[21] **V. Alarcon-Aquino, J. A. Barria.** Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2006, Vol. 36, No. 2, 208–220.

[22] **S. J. Yoo, J. B. Park, Y. H. Choi.** Direct adaptive control using self recurrent wavelet neural network via adaptive learning rates for stable path tracking of mobile robots. In: *IEEE Proceedings of the American Control Conference*, 2005, pp. 288–293.

[23] **S. J. Yoo, J. B. Park, Y. H. Choi.** Stable predictive control of chaotic systems using self-recurrent wavelet neural network. *International Journal of Control, Automation, and Systems*, 2005, Vol. 3, No. 1, 43–55.

Received June 2013.