Neural Network Based Intrusion Detection System for Critical Infrastructures

Ondrej Linda, Todd Vollmer, Milos Manic, *Member, IEEE*

***Abstract*— Resiliency and security in control systems such as SCADA and Nuclear plant's in today's world of hackers and malware are a relevant concern. Computer systems used within critical infrastructures to control physical functions are not immune to the threat of cyber attacks and may be potentially vulnerable. Tailoring an intrusion detection system to the specifics of critical infrastructures can significantly improve the security of such systems. The IDS-NNM – Intrusion Detection System using Neural Network based Modeling, is presented in this paper. The main contributions of this work are: 1) the use and analyses of real network data (data recorded from an existing critical infrastructure); 2) the development of a specific window based feature extraction technique; 3) the construction of training dataset using randomly generated intrusion vectors; 4) the use of a combination of two neural network learning algorithms – the Error-Back Propagation and Levenberg-Marquardt, for normal behavior modeling. The presented algorithm was evaluated on previously unseen network data. The IDS-NNM algorithm proved to be capable of capturing all intrusion attempts presented in the network communication while not generating any false alerts.**

***Index Terms*— Anomaly Intrusion Detection System, Neural Network, Control System**

I. Introduction

CRITICAL infrastructure control systems are often composed of interconnected computer-based systems exchanging crucial information via the computer network. These critical infrastructures, which are the focus of increased security, can be found in systems such as SCADA or nuclear plants [1], [2]. Compromising such a system with intrusion attacks can lead not only to high financial loses but, more importantly, to the endangerment of public safety. The danger is even higher considering that critical infrastructures are not immune to these threats and that they may be potentially more vulnerable than common information technology systems. Hence intrusion protection for critical infrastructures is an obvious need.

Having a system performing predefined legal tasks, an intrusion can be defined as anything that differs from the allowed operations and was in most cases generated with the intention of compromising or misusing the information system. Intrusion detection system (IDS) aims at detecting and tracing such an inappropriate, incorrect, illegal or anomalous activity within the computer network.

The idea of intrusion detection goes back to 1980 and an early intrusion detection model was proposed in 1987 by Dennning [3], [4]. In general there are two kinds of IDS; anomaly detection and signature based detection systems [5].

A database of known and labeled intrusion instances is needed in order for the signature based IDS to work correctly. [5]-[8]. Each instance belongs either to a normal or to an intrusion class. The system is very powerful in recognizing intrusion attacks that match previously seen signatures. The main drawback of a signature based IDS is that dynamically changing intrusion attacks with previously unseen signatures will deceive the system and generate high number of false negatives.

An anomaly IDS seeks deviations from the learned model of normal behavior [9], [11]. The system assumes very little about the features of the future intrusion instances. It builds a representative model exclusively based on the previously collected normal behavior. The system is capable of detecting novel and dynamically changing intrusion instances, when these are distinctively different from the model of normal behavior. Unfortunately, any normal acceptable behavior not included in the training set will not match the model and will generate a false positive. The anomaly intrusion detection approach is adopted in this work.

In this paper a specific window based attribute extraction technique is derived from the analyses of real network data recorded in an existing critical infrastructure. Extracted window based feature vectors capture accurately the trends and the time series nature of the packet stream. A specific combination of two neural network learning algorithms, the Error Back-Propagation and the Levenberg-Marquardt algorithm, is used to train an artificial neural network to model the boundaries of the clusters of recorded normal behavior [12]-[14]. It is shown that the training dataset, consisting of a combination of recorded normal instances and artificially generated intrusion instances, successfully guides the neural network towards learning the complex and irregular cluster boundary in a multidimensional space. The performance of the system is tested on unseen network data containing various intrusion attacks.

The rest of the paper is organized as follows. The network communication within a critical infrastructure is analyzed in section II. Section III gives a description of the extracted window based feature vector. Section IV introduces the IDS-NNM algorithm. Section V presents the achieved experimental results followed by the conclusion given in section VI.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

II. Network Data Acquisition And Analyses

One of the main contributions of this paper is the use and analyses of real network data recorded from an existing infrastructure. The collected data consists of representative samples of normal network behavior, actual intrusion sets as well as data acquired from intrusion attempts. This section describes the data acquisition setup and provides an analysis of the information directly obtainable from packet headers.

A. Network Data Acquisition

Critical infrastructure control systems may consist of interconnected Programmable Logic Controls (PLC) hardware units [15]. An Allen Bradley PLC 5 controller attached to an Ethernet network was used as the testbed for the data acquisition [16]. The PLC controller was connected to a control PC station through an Ethernet hub. The hub is an entry point into the network for data acquisition and intrusion generation. Through this hub the simulated intrusion attempts were generated and the network behavior was recorded. The PLC unit was responsible for controlling valves in a fluid flow structure system. The intrusion attempts were generated using software tools Nmap, Nessus, and MetaSploit [17]-[19]. A diagram of the data acquisition system is shown in Fig. 1.

While the intrusions were artificially generated, they represented a valid estimation of the actual real intrusion attempts that might be experienced by the critical infrastructure. However only confronting the proposed algorithm with these real intrusions can prove its performance and it is a subject of future research and experiments.

B. Network Data Analysis

The packet header is an important source of information describing the network traffic. Attributes from different network layers contain information about the origin of the packet, its target, purpose and function. Examples of attributes extracted from the collected data are: the frame number, the time of recording, the time interval from the previous packet, the sequence number, the acknowledgement number, the protocol type, the window size, data length, the flags code, the source address, or the destination address.

The analyses of the recorded network data showed very regular and stationary patterns of communication. Control and monitoring information was exchanged between the PLC and

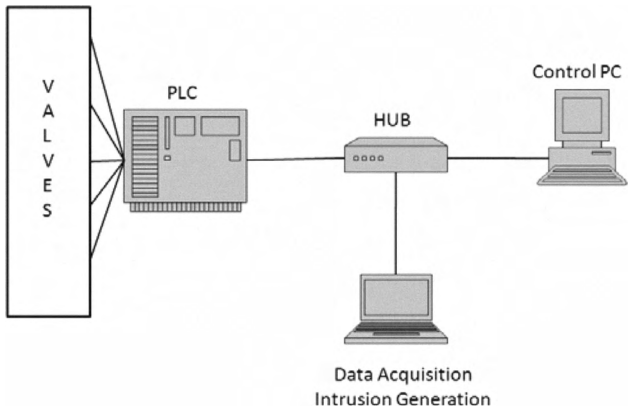

Fig. 1. Network data acquisition setup. A PLC is connected through a hub to the control PC station using an Ethernet network.

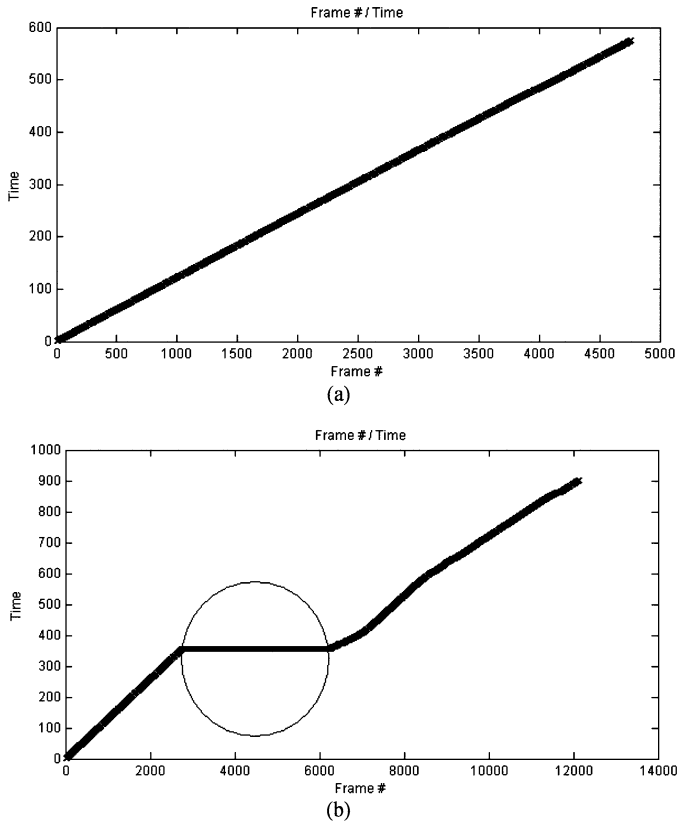

Fig. 2. Plot of the time of recording attribute as a function of the packet frame number during normal communication (a) and during an intrusion (b).

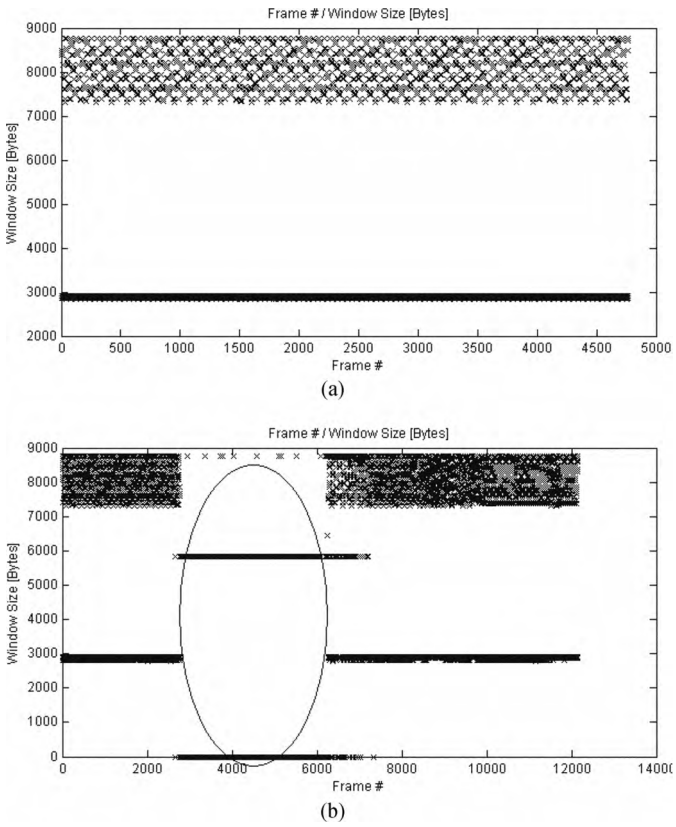

Fig. 3. Plot of the window size attribute as a function of the packet frame number during normal communication (a) and during an intrusion (b).

the control PC at almost a constant rate with very little deviation. Examples of the regular communication pattern are shown in Fig. 2(a) and Fig. 3(a). From Fig. 2(a) the constant speed of the communication is apparent. In Fig. 3(a), two separate streams of packets from the PLC and the control PC can be clearly identified.

Further analysis reveals the clear visibility of recorded intrusion attempts. The simulated intrusion attempts as well as the response of the PLC to these attacks are significantly different from the regular pattern of the normal communication.

Fig. 2(b) and Fig. 3(b) show examples of network communication containing intrusion attacks. Plotting the time of recording and the window size attributes as a functions of the frame number show irregularities that do not match with the regular stream of normal communication. Thus the intrusion attempt is clearly identified. In Fig. 2(b) and 3(b) the intrusion attempt is circled. However, it is important to note that other intrusion examples were not so significantly different from the normal behavior.

It can be observed that the packet headers carry sufficient information to differentiate the normal behavior from an intrusion attempt.

III. Window Based Feature Extraction

As demonstrated in Fig. 2 and Fig. 3, the stream of packets can be described as a time series. Recurrent artificial neural networks are suitable for time-series prediction based intrusion detection [10], [16]. However, a specific window based attribute extraction approach was adopted in this paper.

The time series nature of the packet stream is captured in a single description vector by calculating the statistical features of a limited number of neighboring packets. A window of specified length β is being shifted over the packet stream. At each position, a window based feature vector $\vec{r}_j$ is computed from all the packets $\vec{v}_i$ currently in the window. Consequently, the window is being shifted by one packet forward in the time-sequential ordering of packets. The process of window based feature extraction is illustrated in Fig. 4. In the figure, the new window based feature vector $\vec{r}_j$ is computed based on the attribute extraction from packets $\vec{v}_{i+2} \ldots \vec{v}_{k+1}$ located in the window.

The list of extracted window based attributes is as follows: the number of IP addresses in the window, the maximum and minimum number of packets per single IP, the average interval between packets, the time length of the whole

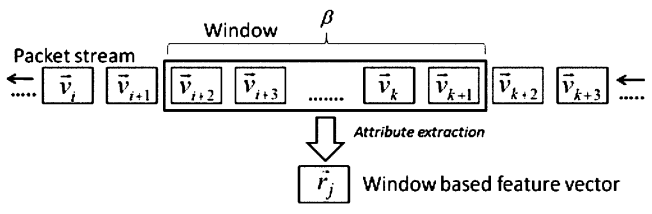

Fig. 4. Window based feature extraction process.

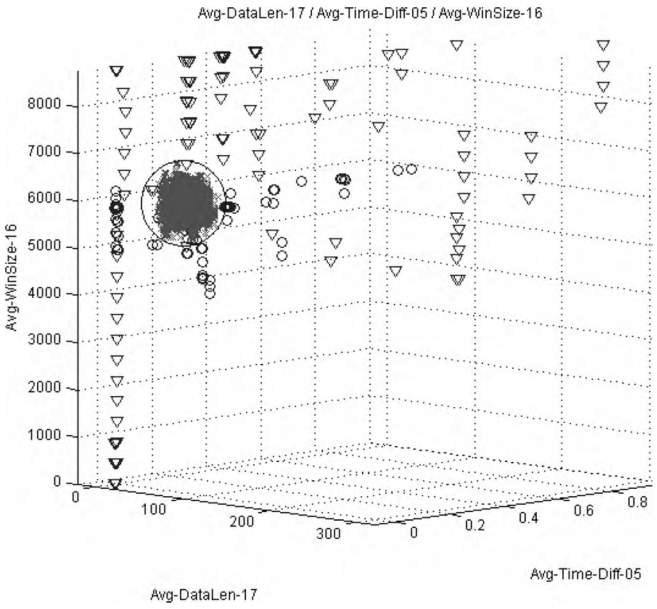

Fig. 5. The network traffic description using the window based attributes.

window, the data speed, the number of protocols in the window, the maximum and minimum number of packets per protocol, the number of flag codes, the maximum and minimum number of packets per flag code, the number of packets with window size attribute set to 0, the number of packets with data length attribute set to 0, the average value of the window size attribute, and the average value of the data length attribute.

These window based attributes were empirically derived based on the analyses of the recorded network traffic and the motivation to most accurately capture the time series nature of the packet stream. Fig. 5 demonstrates the network traffic description using these attributes. Plotted are the instances of normal behavior ($\times$), the intrusion attempts ($\circ$) and the anomalous response of the PLC (∇) respectively. The cluster of the normal network behavior (circled) can be identified surrounded by the anomaly instances.

IV. The IDS-NNM Algorithm

A. Neural Network as a Cluster Boundary Modeling Tool

Clustering constitutes a traditional approach to intrusion detection [5], [21], [22]. The most common problems of clustering techniques are: how to define the number of clusters beforehand; how to initialize of the center of gravities (COGs) of clusters; and how to choose the maximum radius of clusters. Inappropriate choice of these parameters may result in a low performance of the algorithm. Additionally, centroid-based clustering techniques describe a cluster by its COG and by the farthest pattern distance from the COG [23]. Thus the clusters have shapes of hyperspheres in the given input space. This is insufficient for constructing the complex and irregular shapes of clusters in multidimensional spaces.

Artificial neural networks (ANNs) overcome the previously mentioned issues by their inherent capability of constructing

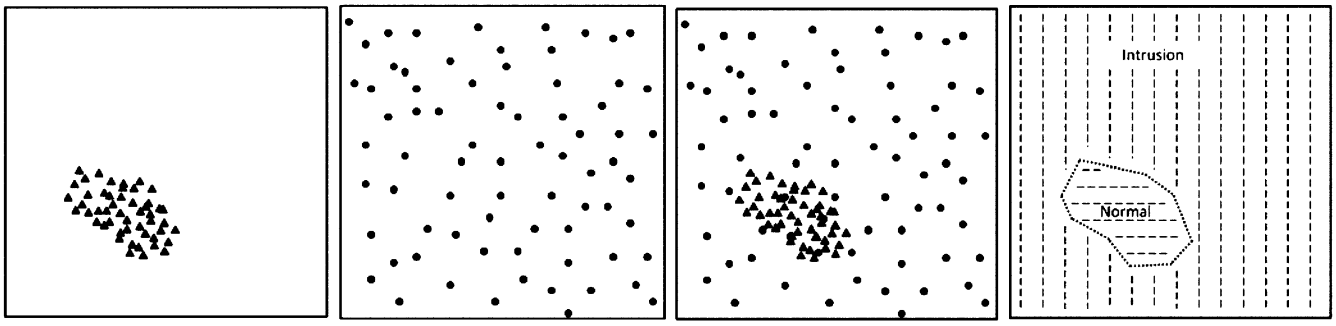

Fig. 6. Illustrative example of the training set construction and the cluster boundary modeled by the neural network. Recorded instances of normal behavior (a) and simulated intrusion instances (b) are combined together into a training set (c). The classification function and the cluster boundary (dotted line) is modeled by the neural network during the supervised learning process (d).

boundaries between classes of irregular and complex shapes in highly dimensional space. The presented IDS-NNM algorithm leverages this capability of the ANNs to accurately model the boundary of the cluster of normal behavior instances. The cluster is modeled by a feed-forward neural network trained in a supervised manner with a specific combination of two learning algorithms, the Error Back-Propagation (EBP) and the Levenberg-Marquardt (LM) learning rule [19]-[21].

Hence, the proposed methodology does not require any upfront knowledge on the number of clusters or their radii. Instead, the ANN is used to define the exact boundary of the normal behavior class. Also, the ANN works as a classifier, directly classifying the current input vector. This proves to be superior to other clustering techniques, where typically the nearest COG has to be found and the decision made based on a heuristically determined threshold.

B. The IDS-NNM Algorithm

The IDS-NNM algorithm consists of two main phases – the specific training set construction and the neural network training process. The trained neural network is applied in the network communication system to detect intrusion attempts.

During the supervised training process the neural network has to be confronted with instances of both normal and intrusion classes. However, in case of an anomaly IDS, future intrusion data vectors are unknown at the time of training. It is only assumed that they will be different from the pattern of the recorded normal behavior.

Hence in the first phase of the IDS-NNM algorithm, the intrusion instances are randomly created in the attribute space. Since the real intrusion vectors are unknown ahead, they will be uniformly generated within the whole attribute space. This newly generated intrusion vector dataset is combined with the recorded normal behavior. Fig. 6(a) – 6(c) illustrate the construction of the training dataset.

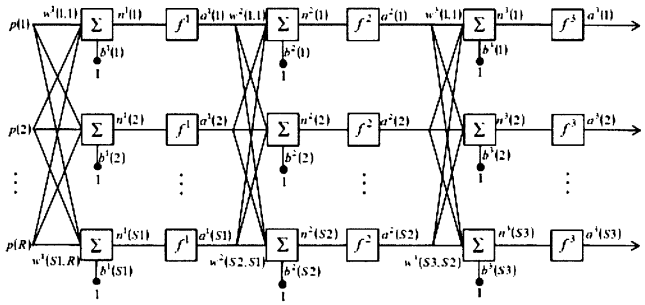

Fig. 7. Three-layer feedforward neural network.

In the second phase of the IDS-NNM algorithm, a feedforward neural network is trained using a specific combination of the Error Back-Propagation and the Levenberg-Marquardt algorithm [19-21]. An example of a three-layer feedforward neural network is shown in Fig. 7.

The output of the input layer is directly determined by the input vector $\vec{p}$:

$$\vec{a}^0 = \vec{p} \tag{1}$$

The net input of neuron i in layer $k+1$ is calculated as:

$$n^{k+1}(i) = \sum_{j=1}^{Sk} w^{k+1}(i,j)\,a^k(j) + b^{k+1}(i) \tag{2}$$

Here Sk denotes the number of neurons in layer k, $w^{k+1}(i,j)$ is the weight of the connection from neuron j in layer k, $b^{k+1}(i)$ is the bias of neuron i and $a^k(j)$ is the output from neuron j in layer k.

The output of neuron i in layer $k+1$ is:

$$a^{k+1}(i) = f^{k+1}\left(n^{k+1}(i)\right) \tag{3}$$

Here f^{k+1} is the activation function of neuron i.

For an L layer neural network, the task of the LM algorithm is to minimize the total error:

$$E = \sum_{p=1}^{P}\sum_{m=1}^{M}\left(d_{pm} - a_{pm}^{L}\right)^2 \tag{4}$$

which can be reduced to:

$$E = \sum_{p=1}^{P}\sum_{m=1}^{M}\left(e_{pm}\right)^2 \tag{5}$$

Here P and M are the number of patterns and the number of outputs respectively, and d_{pm} denotes the desired output.

The weight update rule for the LM algorithm is derived from the Newton's method and written as:

$$\Delta w = A^{-1} g \tag{6}$$

Here A and g are the Hessian and the gradient respectively. For the error function E, which is a sum of squares, the Hessian and gradient can be computed as follows:

$$A \cong 2J^T J \tag{7}$$

$$g = 2J^T \vec{e} \tag{8}$$

Here $\vec{e}$ constitutes the error vector and J is the Jacobian of the partial derivative of error with respect to the weights. The Jacobian matrix can be computed by a modified EBP algorithm [21]. The matrix form of the Hessian and the gradient is written as:

$$A = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_2 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_n \partial w_1} \\ \frac{\partial^2 E}{\partial w_1 \partial w_2} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_n \partial w_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_1 \partial w_n} & \frac{\partial^2 E}{\partial w_2 \partial w_n} & \cdots & \frac{\partial^2 E}{\partial w_n^2} \end{bmatrix} \text{ and } g = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \\ \vdots \\ \frac{\partial E}{\partial w_n} \end{bmatrix} \tag{9}$$

The LM method solves the problem with ill-defined Jacobian matrix by introducing an identity matrix I and learning parameter μ. The LM weight update rule is defined as:

$$\Delta \vec{w} = \left[J^T J + \mu I \right]^{-1} J^T \vec{e} \tag{10}$$

For $\mu = 0$ the LM becomes the Guass-Newton method, whereas for larger values of μ the algorithm is reduced to the steepest decent algorithm. Initially μ is set to *0.001*. If the total error (5) increases, μ is multiplied by 10. In case of error reduction the learning parameter is divided by 10.

Based on the constructed training dataset, the training of the neural network is driven by two assumptions:

1) The intrusions can appear anywhere in the attribute space (including within the cluster of normal behavior);

2) There is a cluster of normal behavior somewhere in the attribute space.

By attempting to minimize the classification error, the training algorithm eventually finds the boundary of the normal behavior class. Anything located outside of the class is therefore considered an intrusion. Fig. 6(d) describes the learned classification function.

The steps of the IDS-NNM algorithm are as follows:

Step 1.1: Construct an ordered sequence S_T of attribute vectors $\vec{v}_i$ using the information from packet headers. The vectors are order time-sequentially:

$$S_T = \{\vec{v}_0, \vec{v}_1, ..., \vec{v}_N\} \tag{11}$$

Here, $\vec{v}_0$ and $\vec{v}_N$ are the first and the last recorded packets in the sequence, respectively.

Step 1.2: Extract sequence S_W of window based feature vectors $\vec{r}_j$ from sequence S_T. This extraction of window based attributes can be described as:

$$f(\vec{v}_i, \vec{v}_{i+1}, ..., \vec{v}_{i+\beta-1}) \rightarrow \vec{r}_j, \quad i, j \in \{0, 1, ..., N - \beta + 1\} \tag{12}$$

Where β denotes the length of the window.

Step 1.3: Create set S_W^* of normal behavior training instances by assigning each feature vector $\vec{r}_j$ class label l_{Norm}.

$$S_W^* = \{(\vec{r}_j, l_{Norm})\}_{j=1,2,...,N-\beta+1} \tag{13}$$

Step 1.4: Create randomly generated set I_W of simulated intrusion vectors uniformly distributed over the window based attribute space.

$$I_W = \{\vec{r}_0, \vec{r}_1, ..., \vec{r}_M\} \tag{14}$$

Where M is the number of generated intrusion vectors.

Step 1.5: Create set I_W^* of the intrusion training instances by assigning each feature vector $\vec{r}_k$ class label l_{Intr}.

$$I_W^* = \{(\vec{r}_k, l_{Intr})\}_{k=1,2,...,M} \tag{15}$$

Step 1.6: Combine sets S_W^* and I_W^* into a single training dataset T:

$$T = S_W^* \cup I_W^* \tag{16}$$

Step 2.1: Propagate the training dataset T to the output of the neural network using (1), (2) and (3).

Step 2.2: Using the modified EBP compute the Jacobian matrix.

Step 2.3: Calculate the weight update vector $\Delta \vec{w}$ by solving (10).

Step 2.4: Update the network weights and the learning parameter μ:

Step 2.6: If predefined convergence criteria is not met, go to step 2.1.

V. Experimental Results

This section presents the experimental results. The suitable architecture of the neural network as well as the importance of using only the relevant attributes is discussed and demonstrated. The performance is evaluated on the recorded real network data.

A. Testing Datasets

The data acquisition testbed is shown in Fig. 1. Software tools Nmap, Nessus and MetaSploit were used to generate various intrusion attacks. Even though only simulated, the intrusion attacks represented representative samples of expected real intrusion challenges. Five datasets were recorded directly from the network communication. Each

dataset consists of approximately 20000 packets. To accurately model the normal network behavior, an additional data set of purely normal network behavior was recorded. 100000 randomly simulated intrusion vectors were generated for all the experiments.

B. Intrusion Detection Evaluation

The performance of the IDS-NNM algorithm was tested on the recorded network traffic datasets. It was measured by the detection rate and the false positive rate. The detection rate represents the ratio between the correctly identified intrusion attacks and the overall number of intrusions in the dataset:

$$Detect\_Rate = \frac{N_{Detected\_Intrusion}}{N_{All\_Intrusions}} \quad [\%] \qquad (15)$$

The false positive rate calculates the ratio between the number of instances of normal behavior falsely marked as an intrusion and the overall number instances of normal behavior:

$$False\_Positive = \frac{N_{False\_Normal}}{N_{All\_Normal}} \quad [\%] \qquad (16)$$

The classification function of the system can be adjusted by setting a sensitivity threshold. When the output value of the

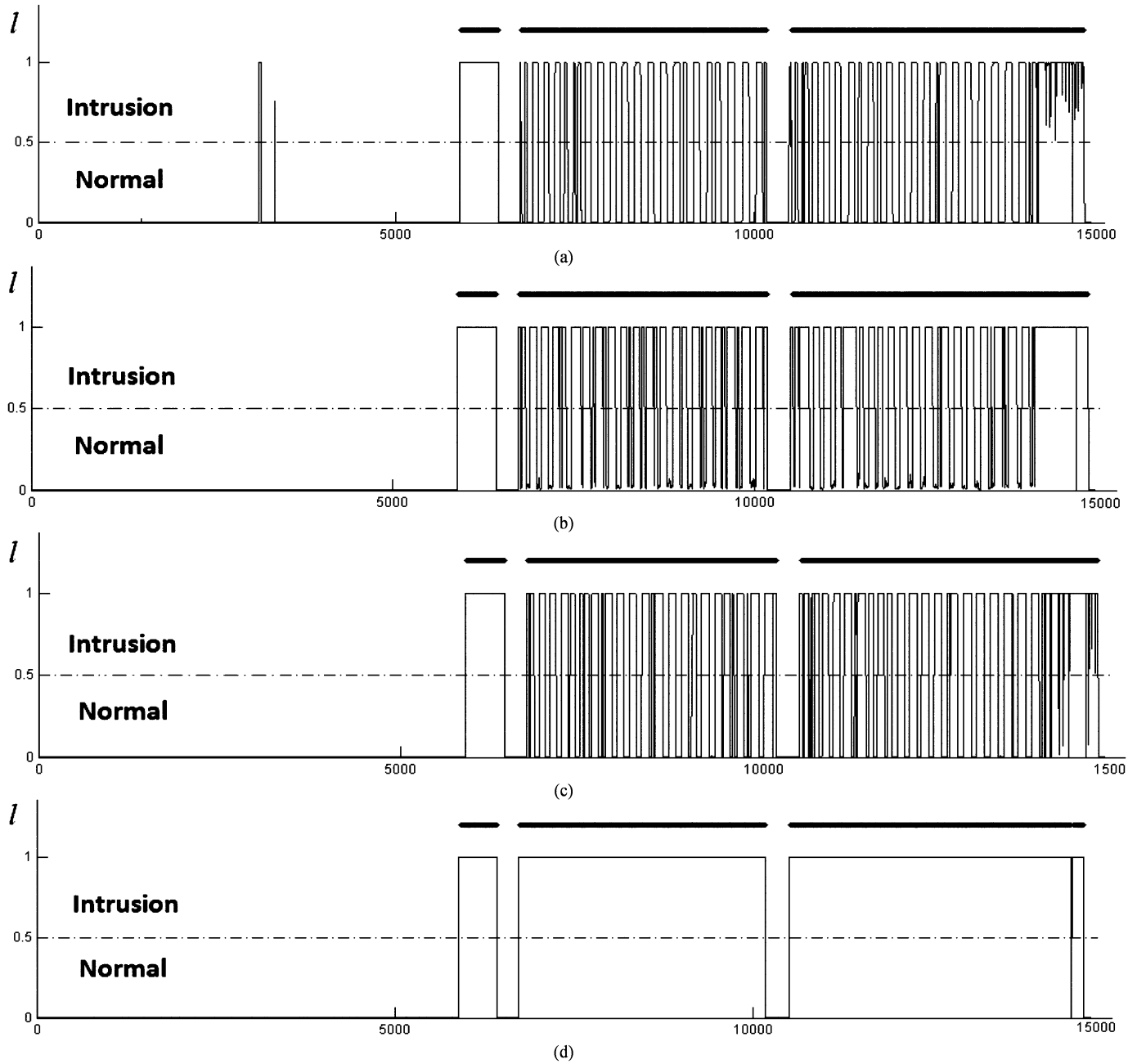

Fig. 8. Intrusion detection of datasets 1 using 1 hidden layer and 16 attributes (a), 1 hidden layer and 8 attributes (b), 2 hidden layers and 16 attributes (c) and 2 hidden layers and 8 attributes (d).

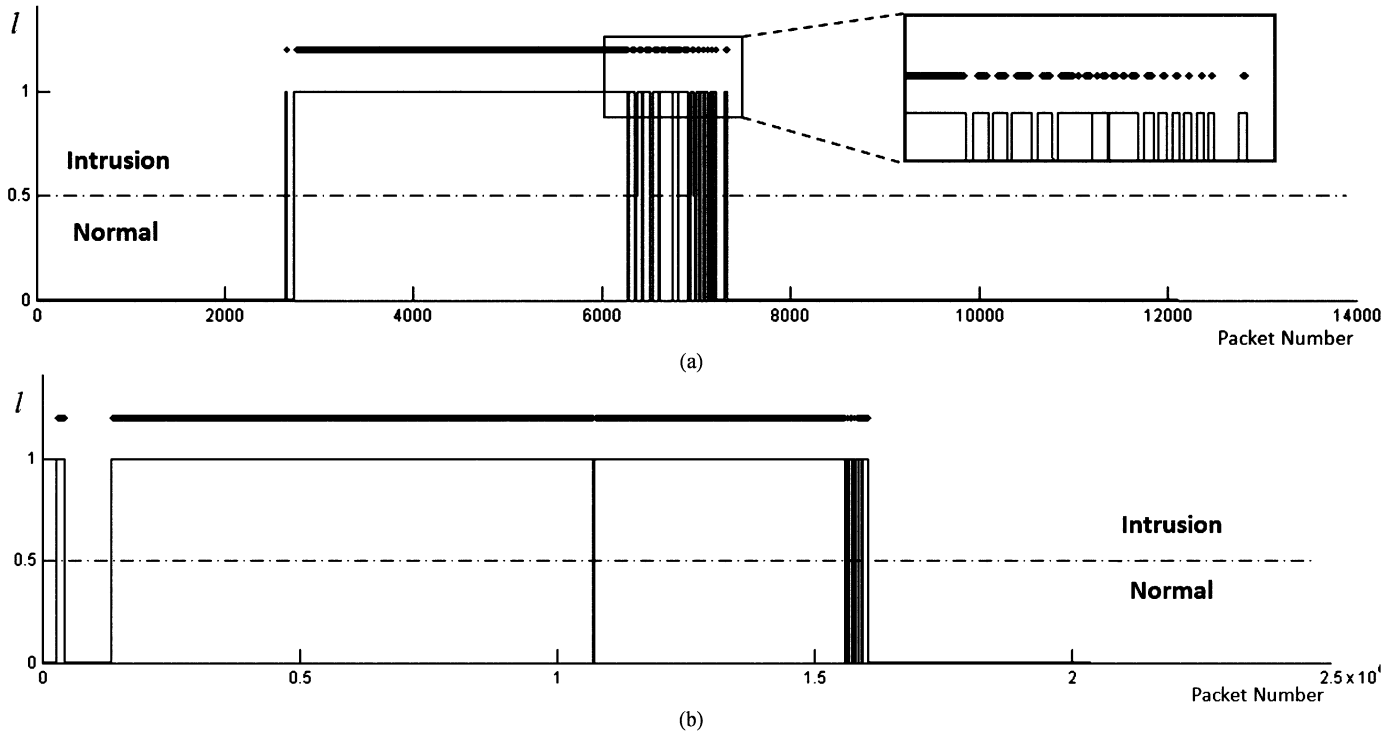

Fig. 9. Intrusion detection of datasets 2 (a) and 3 (b) using the presented IDS-NNM algorithm.

TABLE I
MOST SIGNIFICANT WINDOW BASED ATTRIBUTES

Num. of IP addresses	Num. packets with 0 win. size
Avg. interval between packets	Num. packets with 0 data lenght
Num. of protocols	Average window size
Num. of flag codes	Average data length

neural network is above the sensitivity threshold, the input vector is marked as an intrusion. Otherwise it is denoted as a normal network behavior.

The size of the window for the window based feature extraction was set to 20 packets. This value is a compromise between having enough packets to accurately compute the statistical properties of the packet stream and having too many packets in the window to hide short intrusion attacks.

C. Relevant Attributes Selection and ANN Architecture

Each of the constructed window based attributes has a different importance to the classification task. Reducing the dimensionality of the problem to the most relevant attributes only is critical for two main reasons: 1) the classification task is generally easier in spaces with lower dimensionality and using only the relevant attributes; 2) the needed number of randomly generated intrusion instances grows exponentially with the dimensionality of the problem.

The experimental leave-one-out approach was used to identify the eight most significant window based attributes. They are listed in Table I.

The architecture of the used feed-forward neural network also has to reflect the complexity of the problem. Several architectures were tested in order to locate the optimal one. The identified architecture consisted of two hidden layers with 10 and 6 neurons in first and the second layer respectively and one output neuron.

TABLE II
PERFORMANCE OF THE IDS-NNC ALGORITHM ON DATASET 2

Data Set	Detection Rate	False Positive
1 layer, 16 inputs	66.063%	0.378%
1 layer, 8 inputs	85.081%	0%
2 layers, 16 inputs	78.643%	0%
2 layers, 8 inputs	100%	0%

TABLE III
PERFORMANCE OF THE IDS-NNC ALGORITHM

Data Set	Detection Rate	False Positive
Testing Set 1	100%	0%
Testing Set 2	100%	0%
Testing Set 3	100%	0%
Testing Set 4	100%	0%
Testing Set 5	100%	0%

Fig. 8 demonstrates the effect of using relevant attributes and the optimal network architecture on the performance of the system. The classification performed by the network can be compared to the true occurrence of the intrusion attempts, marked with a bold line. The classification performance of particular setups on one of the recorded datasets is summarized in Table II.

Fig. 8(a) shows the performance of neural network with only 1 hidden layer with 10 neurons trained on all the 16 attributes. It resulted in a quite poor detection rate (66.08%) and several false positives. Reducing the number of attributes to the eight most significant ones and the same network architecture substantially improved the performance (85.08%) and no false positives were generated as shown in Fig. 8(b).

Similarly, expanding the neural network structure into 2 hidden layers and using all 16 attributes led to an improvement of the detection rate (78.64%). Finally, using the expanded network architecture with 2 hidden layers and training on the 8 most relevant attributes the optimal performance of the system (100%) was achieved as demonstrated in Fig. 8(d).

Table III summarizes the experimental results achieved on all the 5 recorded datasets containing the intrusion attempts. Further, Fig. 9 shows another two examples of the system's performance on datasets 2 and 3. The zoomed section of Fig. 9(a) shows that the neural network correctly identifies even short intrusion attempts.

VI. Conclusion And Further Work

This paper presented a novel intrusion detection system tailored to the specifics of critical infrastructures. The main contributions of this work were: 1) the use and analyses of real network data; 2) the development of specific window based feature extraction technique; 3) the construction of training dataset using randomly generated intrusion vectors; 4) the use of a specific combination of two neural network learning algorithms – the Error-Back Propagation and Levenberg-Marquardt, for normal behavior modeling.

The IDS-NNM - the Intrusion Detection System using Neural Network based Modeling algorithm - achieved a perfect detection rate while generating no false positives on previously unseen testing data. The presented experimental results illustrated the ability of the system to detect long intrusion attacks as well as short intrusion attempts consisting only of several packets. This demonstrated the correctness of the presented window based feature extraction mechanism as well as the power and robustness of the artificial neural network as a cluster boundary modeling tool. Further, the importance of identifying the relevant attributes and using the suitable ANN architecture was demostrated.

Further research is intended in the area of extracting additional significant features of the network traffic as well as generating other different types of intrusions to test the implemented system. Furthermore, the performance of the algorithm as a function of the length of the window used for feature extraction will be addressed.

Acknowledgment

The authors would like to thank the Idaho National Laboratory and the University of Idaho Nuclear Engineering Program for providing support for this project.

References

[1] D. Yang, A. Usynin, J. W. Hines, "Anomaly-Based Intrusion Detection for SCADA Systems", *5th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05)* , Albuquerque, NM, Nov 12-16, 2006.

[2] H. S. Kim, J. M. Lee, T. Park, W. H. Kwon, "Design of networks for distributed digital control systems in nuclear power plants", *Intl. Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies (NPIC&HMIT 2000)*, Washington, DC, November 2000.

[3] J. P. Anderson, *Computer security threat monitoring and surveillance*, Technical report, James P. Anderson Co, 1980.

[4] D. E. Denning, "An Intrusion Detection Model", IEEE Transactions on Software Engineering,Vol. SE-13, February 1987, pp. 222-232.

[5] S. Zhong, T. Khoshgoftaar, N. Seliya, "Clustering-based network intrusion detection", In *Intl. Journal of Reliability, Quality and Safety*, Vol. 14, No. 2, 2007, pp. 169-187.

[6] K. Ilgun, R. A. Kemmerer, P. A. Porras, "State transition analyses: A rule-based intrusion detection system", *IEEE Transaction on Software Engineering*, 21(3), March 1995.

[7] G. Stein, B. Chen, A. S. Wu, K. A. Hua, "Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection", in Proceedings of the 43rd ACM Southeast Conference, Kennesaw, GA, March 2005.

[8] W. Lee, S. Stolfo, P. K. Chan, "Learning patterns from unix process execution traced for intrusion detection", In *Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management*, 1997.

[9] J. Ryan, M. Llin, R. Miikkulainen, "Intrusion Detection with Neural Networks", In *Advances in Neural Information Precessing Systems 10*, Cambridge, MA, MIT Press, 1998.

[10] A. K. Gosh, A. Schwartzbard, M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection", In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, April 1999, pp. 51-62.

[11] A. K. Gosh, J. Wanken, F. Charron, "Detecting anomalous and unknown intrusions against programs", In proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC'98), December 1998.

[12] P. J. Werbos, *The Roots of Backpropagation*, New York: Johns Wiley & Sons, 1994.

[13] D. Marquardt, "An algorithm for least squares estimation of non-linear parameters," J. Soc. Ind. Appl. Math., pp.431-441, 1963.

[14] M. Hagan, M. Menhaj, "Training feedforward networks with the Marquardt algorithm," IEEE Transaction on Neural Networks, vol. 5, no. 6, pp. 989-993, 1994.

[15] Dana A. Shea, "Critical Infrastructure: Control Systems and the Terrorist Threat," Report for Congress RL31534, February, 2003.

[16] Allan Bradley PLC 5 Controller: http://www.ab.com/programmablecontrol/plc/pclsystem/index.html

[17] Nmap – "Network Mapper": http://nmap.org

[18] Nessus: http://www.nessus.org.org/nessus/

[19] The Metasploit Project: http://www.metasploit.com/home

[20] H. Debar, B Dorizzi, "An Application of a Recurrent Network to an Intrusion Detection System", In Proceedings of the International Joint Conference on Neural Networks, pp. 78-83.

[21] Q. Wang, V. Mehalooikonomou, "A Clustering Algorithm for Intrusion Detection," in SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, Orlando, Florida, USA, 2005.

[22] L. Portnoy, E. Eskin, S. Solfo, "Intrusion detection with unlabeled data using clustering," Proc. Of ACM CSS Workshop on Data Mining Applied Security, Philadelphia, PA, November 5-8, 2001.

[23] I. H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 2005.