A Neural Network Classifier based Decision Support System (NNCDSS) for Network Intrusion Detection and Response

Tich Phuoc Tran, Dong Fang Wang, Shuoyu Chen, Marshall Tolentino, Jie Lu

Faculty of Information Technology University of Technology, Sydney
P.O. Box 123, Broadway, NSW, 2007, Australia
Email: {tiptran, dfwang, shychen, mtolenti, jielu}@it.uts.edu.au

Abstract

This paper introduces an innovative design of a classifier based decision support component for an intrusion detection system. In particular, this model uses an emerging semi-parametric learning algorithm called Modified Probabilistic Neural Network to capture both attacks' signatures as well as normal system usage behaviours. The statistics from this detection system is then assessed by an Expert System to generate a risk level and recommended actions in response to the detected anomalies. Extensive experimental outcome shows that the proposed model outperforms existing methods in terms of detection accuracy, model robustness and computational cost.

Keywords: intrusion detection, neural network, expert systems, decision support systems.

1. Introduction

Communication on the Internet provides convenience and benefits such as shortening effective geographical distances and efficiently sharing information. However, this poses a major threat such that the system may be compromised by intruders. Potential attacks can result in the company's loss of confidentiality, authenticity, credibility and the integrity to its customers. With respect to a company's reputation, negative publicity (i.e. security vulnerabilities in applications) impacts potential sales and plays an essential role in adversely decreasing customer's confidence in the company's products and as well as its service. In addition, fixing a ruined public image and gaining back lost trust amongst customers are considered to be more expensive than actually building an application system that mitigates security risk breaches. Therefore, companies should prioritize and focus on their IT security strategies and budget on the protection of their network and physical access controls. To protect computer network, different defensive techniques are implemented. An emerging technology in this field is Intrusion Detection System (IDS). An IDS is defined as a protection system that monitors the network traffic and computing usage behaviour to detect any suspicious activity that could compromise the computer networks [1]. Because cyber attacks inevitably manifest themselves in host audit data and/or in network traffic data [2], an IDS can be used to monitor and analyse data from those sources to detect potential attacks. Examples of some popular commercial IDS products include Snort, NetRanger, RealSecure, CISCO IOS IDS and Omniguard. One of the most challenging issues of these existing IDS is the lack of user friendliness which reduces the usability and productivity of the system.

Most of the IDS in the market are signature-based or misuse-based detection systems in which the detection process involves monitoring network or system resources and comparing observed data from the sensors to a signature database of known adversarial behaviours [3]. This database needs to be updated frequently to cope with the dynamic and complex nature of network attacks. There is a trend in Network Security whereby Artificial Intelligence (AI) techniques provide an adaptive, automatic detection system with capability of self-learning from available data. For example, Neural networks (NN) can be used to recognise normal user behaviours in anomaly detection systems in which the activities deviating from "normal" operations are detected as attacks. In particular, NN is trained with data sources such as network usage behaviours or command list executed by different users. The resulting classifier then identifies malicious behaviours that do not match its training experience. In the paper [8], an intrusion dataset was used to train a model which is a

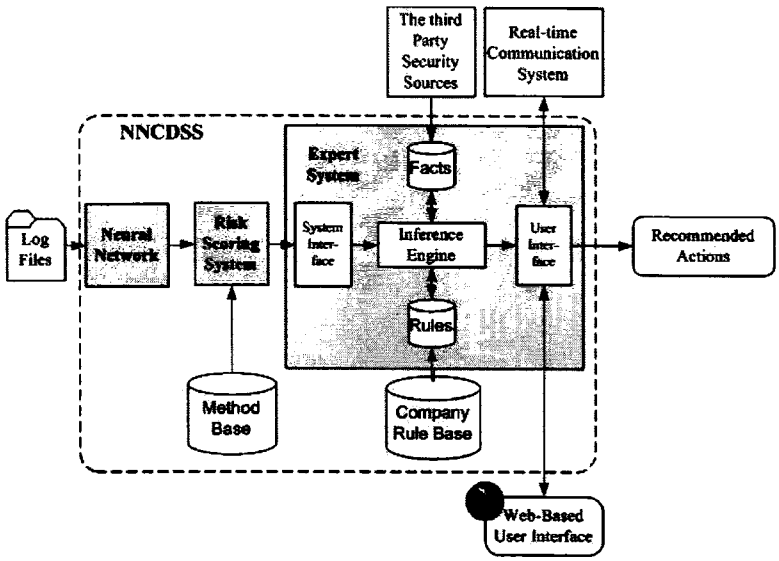

Figure 1 Logical Model of NNCDSS

combination of Self-Organizing Map (SOM) with Resilient Propagation Neural Network (RPROP) for intrusion detection. SOM was used for clustering and visualization while RPROP was used to classify normal patterns and intrusions. This dataset was divided into eight subsets and a SOM was trained with each subset. Neuron weights from the eight resulting SOM were then fed to the three-layer PRPROP neural network. The classification takes place at the third level of PRPROP. Another NN based approach involved a Multi Level Perceptron (MLP) architecture that consists of four layers [9]. This is a misuse detection system which was trained with a dataset collected from the Internet Security Stream.

Though the popularity of the implementation of AI for Network Intrusion Detection, the existing AI algorithms are still far from an acceptable solution for large and complex network data due to some limitations such as low accuracy and high false alarm rates for rare and complicated attacks. In reality, to obtain high accuracy and robustness for a learning model, a sufficient load of data is needed and this requirement normally leads to high computational cost. To overcome those difficulties, this paper has designed a Neural Network Classifier Based Decision Support System (NNCDSS) for network misuse detection. NNCDSS uses NN as a classifier to detect different types of network attacks. The result of this classification is then inputted into an Expert System (ES) to assess the network security risk level and recommend several corrective actions. In order to demonstrate and compare the outcome of this system with other existing methods, a dataset which was provided by The Knowledge Discovery and Data Mining contest in 1999 (KDD-99) will be used as the benchmark.

2. Logical Model Design of NNCDSS

The logical model of NNCDSS is shown in Figure 1. There are 3 major components including a NN, a Risk Scoring System (RSS) and an ES. Firstly, NN is trained with some log files retrieved from the specific log servers. These logs contain general information of network connections such as source destination address, protocol and port numbers. The resulting NN is capable of classifying unseen data into different classes (normal or attacks). The classification result from this is then passed to the RSS to compute the Risk Score associated with a network status. This process also uses some methods and formulas provided by the Method Base.

After the score (which represents the risk of the network) has been calculated, it is compared against the risk thresholds configured by an ES. This ES has a system interface which handles the seamless data communication between sub systems. The Knowledge Base of the ES contains rules and facts. These rules are provided by the company rule base such as organizational security policies, company VPN information; while the public standard security information (facts) such as computer virus base, malicious web sites list and spam origination are collected from third party security resources such as Symantec and MacAfee. These sources can be updated

dynamically by security expert communities. From these rules and facts, the Inference Engine correlates information and identifies the Risk Level from the computed Risk Score. Base on different risk levels, the ES will then suggest several corrective actions to terminate existing intrusions or prevent future attacks from ever occurring. The risk level and recommended actions will be displayed by a user interface in several graphical forms (such as bar chart or pie chart.) This interface can be accessed by the administrator through a secure internet connection with authentication and secured protocol or via a mobile facility such as PDAs and mobile phones. The accessibility of the system ensures that the relevant authorities are notified as soon as a security issue is identified, regardless of the working hours. In other words, the security violations will be taken into a great account in a timely manner. The later sections of the report examine the details of the major sub-systems such as the Classification component (NN), the RSS and the recommended generator.

3. Classification Component Using Modified Probabilistic Neural Network (MPNN)

In this paper, the NNCDSS uses MPNN as the core classification method. MPNN is a recently emerging algorithm which was initially introduced by Anthony Zaknich in 1991 [14] for application to general signal processing and pattern recognition problems. MPNN is a generalization of Probabilistic Neural Network (PNN) and is related to General Regression Neural Network (GRNN) classifier [15]. In particular, this method generalises GRNN by assigning the clusters of input vectors rather than each individual training case to radial units. The following is the general form of GRNN which is similar to the equation proposed by Nadaraya and Watson [16]:

$$\hat{y}(\underline{x}) = \frac{\sum_{n=1}^{NV} y_n f_n(\underline{x} - \underline{x}_n, \delta)}{\sum_{n=1}^{NV} f_n(\underline{x} - \underline{x}_n, \delta)} \quad (1)$$

With Gaussian function:

$$f_n(\underline{x}) = \exp\frac{-(\underline{x} - \underline{x}_i)^T (\underline{x} - \underline{x}_i)}{2\delta^2} \quad (2)$$

Where

$\underline{x}$: Input vector (under line refers to vector)

$\underline{x}_n$: All other training vectors in the input space

δ: Single smoothing parameter chosen during network training

y_n: Scalar output related to $\underline{x}_n$

NV: Total number of training vectors

Equation 2 can be rewritten in a more compact form:

$$f_i(d_i, \delta) = \exp\frac{-d^2}{2\delta^2} \quad (3)$$

Where $d_i = \| x - \underline{x}_i \| = \sqrt{(x - \underline{x}_i)^T (x - \underline{x}_i)}$

From Equation 1, each input vector has an associated equal size Gaussian function and a corresponding scalar output. This Gaussian function is then applied on the Euclidian distances of an input vector to all other vectors in the input space. Given an input vector $\underline{x}$, the corresponding output vector $\hat{y}(\underline{x})$ is then computed by dividing the sum of the scalar y_n and Gaussian $f_n(.)$ products by the sum of Gaussian $f_n(.)$ functions.

In many applications, GRNN provides high accuracy [14]. However, it is computationally expensive as well as sensitive to the selection of variances for smoothing functions. In fact, GRNN incorporates each and every training example $\{\underline{x}_i \rightarrow y_i\}$ into its architecture, i.e. all of the training vectors needs to be processed and Gaussian function's parameters such as centers and variance will need to be computed with respect to all other surrounding vectors. In order to overcome this problem, MPNN approximates GRNN by quantizing the data space into clusters and associate a specific weight for each of these clusters.

If there exists a corresponding scalar output y_i for each local region (cluster) which is represented by a center vector $\underline{c}_i$, then a GRNN can be approximated by a MPNN formulated as follow:

$$\hat{y}(\underline{x}) = \frac{\sum_{i=0}^{M} Z_i y_i f_i(\underline{x} - \underline{c}_i, \delta)}{\sum_{i=0}^{M} Z_i f_i(\underline{x} - \underline{c}_i, \delta)} \quad (4)$$

With Gaussian function

Where

$$f_i(\underline{x}) = \exp\frac{-(\underline{x} - \underline{c}_i)^T (\underline{x} - \underline{c}_i)}{2\delta^2}$$

$\underline{c}_i$ = center vector for cluster *i* in the input space

y_i = scalar output related to $\underline{c}_i$

Z_i = number of input vectors x_j within cluster $\underline{c}_i$

δ = single smoothing parameter chosen during network training

M = number of unique centers $\underline{c}_i$

Equation 4 can be seen as the general formulation for both GRNN and MPNN. In another word, GRNN can be computed from this equation by assuming that one input vector is assigned to each cluster ($Z_i = 1$), the y_i are real values (the output space is not quantized), the centre vectors $\underline{c}_i$ are replaced by with individual training vectors x_i and the number of clusters is equal to the number of individual input vectors ($M = NV$).

Comparing Equation 4 and Equation 1 of GRNN, we can find the only difference is that MPNN applies its computation on a smaller number of clusters of input vectors represented by centers vectors $\underline{c}_i$ rather than working with individual input vectors $\underline{x}_n$. This clustering relies on Gaussian characteristics in which summing multiple Gaussian functions is equivalent with having a single Gaussian provided that they are well concentrated (clustered) near the centers in the data space. This idea is formulated as follow:

$$\sum_{i=1}^{Z_j} f_i(\underline{x}-\underline{x}_i,\delta) \approx Z_j f_j(\underline{x}-\underline{c}_j,\delta)$$

4. Risk Scoring and Recommended Actions

In the NNCDSS, the RSS assesses the risk of a particular computer network with respect to the severity of different attack types as well as their frequencies. In particular, each attack type will be associated with a weight. A higher weighted attack is considered to be more dangerous than the lower ones. The NN detects these attacks' occurrences and generate the frequencies for each of them. Given the weights and frequencies for each attack types, the network security status will be evaluated by calculating a risk score which is computed as follows:

Assume that there are ***n*** types of connections (normal connections or some kind of attacks) which can be detected by the IDS. Connection of type ***i*** (i=0 for normal connections, i=1 for Probe, i=2 for DoS, i=3 for U2R and i=4 for R2L) is assigned a weight of W_i and has a frequency of C_i. For each type of connection, the detected frequency is first normalized by dividing this frequency by the largest frequency of all the connection classes:

$$N_i = \frac{C_i}{C_{\max}}$$

In this formula, i is the connection type index, C_i is the original frequency of connection type i, N_i is the normalized frequency. After this normalization, the risk score can be calculated as shown in equation 1.

$$R = \frac{\sum_{i=0}^{n} W_i * N_i}{\sum_{i=0}^{n} N_i} \qquad (5)$$

Where

W_i : weight of connection type i.

R: risk score

From equation 5, R is between 0 and 1. This score is then compared against thresholds, which are configured by the ES to specify the appropriate risk level. Each of the risk levels will correspond to a set of recommended actions. The following diagram shows the relationship of different risk levels to its related recommended actions.

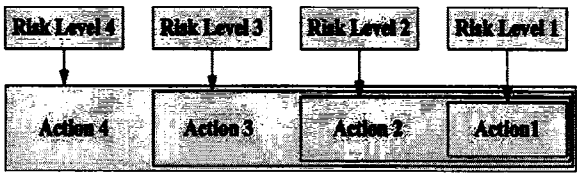

Figure 2 Risk Level and Recommended Actions for NNCDSS

Table 1 Recommended actions

Risk Level	Recommended Actions
1	1. Password changing; Firewall database updating; Apply new system patch; Enhance security awareness
2	2. Filter malicious IP, websites; Block risky ports; Alert administrator as soon as possible
3	3. Implements new network proxy; Apply new security policy
4	4. Backup important data; Need network security experts intervention

As shown in Table 1, a more serious situation, corresponds to higher risk level. In order to keep the integrity when applying higher recommended actions, a lower action must be applied first. For instance, the implementation of the recommended action for risk level 4 must follow the accomplishment of action 3.

5. Experimental analysis

To investigate the advantages of the proposed system, this paper used the dataset provided by MIT Lincoln Laboratory for the Knowledge Discovery & Data Mining in 1999 conference (KDD-99) as a benchmark. In this case, the dataset contains approximately 5 million network connection records and 41 independent features and a label indicating the connection is either normal or has been subjected to a specific attack type [10]. These attacks fall into 4 attack categories, which are: probe, denial of service (DoS), remote to local (R2L) and user to root (U2R) attacks. In particular, probe attacks refer to automatic scanning computer networks to gather useful information for intruders, while a DoS attack is considered to be more serious, as it obstructs the legitimate requests to system resources. The R2L attacks occur when an intruder gains local access to the system by sending packets over networks. Finally, U2R attacks assume that the intruder has a normal user account, which eventually can exploit the system vulnerabilities to gain root user privileges.

The purpose of this experiment is to prove the effectiveness of the proposed NNCDS in a number of applications. In this paper, NNCDS is implemented to detect 5 class labels from the KDD-99 intrusion dataset including Normal, probe, DoS, U2R and R2L. It is then compared against other learning techniques, as well as the KDD-99 winning method. Specifically, the MPNN algorithm is compared with J48 Decision Tree, Multilayer Perceptron (MLP) and GRNN in terms of detection rate (accuracy), training time (model complexity) and model sensitivity (variance). For the RBFNN models, similar model size was chosen with one hidden layer containing 15 hidden nodes. The MLP has a structure of 3 layers with a number of input neurons which is equivalent to the number of input features, 5 hidden neurons and 5 output neurons. The learning models are trained with the full set of attributes. The entire KDD-99 (around 5 million records) is split into a training set of 100,000 records and a testing set of 50,000 records for 5 instances. Some evaluation metrics are selected such as detection rates, false positive rate, training time (indicates the computational requirement) and misclassification cost (according to the cost matrix given by KDD-99). Table 2 below shows the comparison between BMPNN and other learning techniques:

Table2 Performance comparison between different algorithms for KDD-99 dataset

Algorithm for PROBE	**Detection Rate**	**Training time (minutes)**	**Model sensitivity (variance)**
J48 Tree	91.21%	3.7	0.131
MLP	94.35%	21	0.201
GRNN	94.45%	15.7	0.032
MPNN	**94.46%**	**4.1**	**0.031**

In general, MPNN is shown to achieve the lowest generalization variance (0.031), highest accuracy (94.46%), and is considered to be more accurate than GRNN (94.45%). For the training time, MPNN appeared to take minimal time in training the model. Although the Decision Tree has the fastest training time, it was unable to consider all other factors. MPNN, on the other hand, is considered to be the best trade-off between accuracy, stability and computational cost. Lastly, the traditional MLP failed to perform efficiently in this problem, having slow training time and high variance.

Table 3 and Table 4 have displayed the results of the detection rates and false positive rates for each class labels predicted by the BMPNN classifier and the KDD-99 winner.

Table 3 Detection rates (%) of proposed model and the winning method of KDD-99

Model	**Attack Categories**				
	Normal	**Probe**	**DoS**	**U2R**	**R2L**
MPNN	4.15	0.23	0.29	0.04	0.09
KDD99 winner	8.19	0.61	0.32	0	0.01

Table 4 False positive rates (%) of proposed model and the winning method of KDD-99

Model	**Attack Categories**				
	Normal	**Probe**	**DoS**	**U2R**	**R2L**
MPNN	95.92	94.23	96.27	34.88	49.83
KDD99 winner	99.45	83.32	97.12	13.16	8.40

From the above results, the detection rates for Normal and DoS connections are considered to be very comparative. Our method, in this regard, can further improve the detection ability for Probe attacks. More interestingly, the two most difficult detected attacks, U2R and R2L, can be identified efficiently by our model, achieving a result of 34.88% and 49.83%, respectively. Although our success for enhancing detection rates for most of the attack categories

appeared to be satisfactory, the proposed model also embraces some disadvantages in relation to false positive rates. In particular, the false positive rates for U2R and R2L produced by our model are 0.04% and 0.09% respectively, which are higher than those rates generated by the KDD-99 winner (nearly zero false positive rates). In the positive side, the proposed method still outperforms the Normal and Probe classes with significant low false positive rate (4.15% and 0.23% respectively) compared to the KDD-99 entry.

Apart from the above, it is calculated that the misclassification cost associated with our method is 0.1822 which is less than that of KDD-99 winner (0.2331). This may be due to the significant improvement in the accuracy of U2R and R2L detections (these attacks have high cost for misclassification). Though our system results in slightly lower detection rates for Normal and DoS in comparison with the KDD-99 winning entry, our model significantly reduced false positive rates for Normal, DoS, Probe and likewise increased true positive rate for U2R and R2L attacks.

6. Conclusions and Further Study

The issue of network security is becoming more critical to the success of businesses. Therefore, they should take into great account their exposure to different types of risk attacks. An emerging technology in this field is an IDS, which is a system that can detect potential cyber attacks. Although the concept of IDS has several limitations (e.g. user unfriendliness and difficulty in interpreting results), it still plays an essential role in protecting one's networks. Hence, this paper attempted to propose the use of AI techniques such as NN and ES for intrusion detection, which can potentially improve the performance of the current IDS. In particular, MPNN is implemented as a classifier to detect different suspicious activities. The result from this classification process is then passed onto an ES to assess the network risk level and from there, obtain several recommended corrective actions. The extensive experiments from KDD-99 benchmark showed that our proposed model has a higher detection rate, which considerably improved the system's robustness and finally reduced the computational cost. Some features such as real time notification, remote access and system updates from third party sources are also discussed in this paper. Likewise, several motivational factors that businesses should consider when attempting the implementation of IDS in one's enterprise was also presented.

The generic IDS proposed in this paper can be extended to a detection system for the networks whereby computer resources are highly distributed. The relevant literature has highlighted several generic limitations associated with Distributed IDS (DIDS) such as inability to cope with huge amount of data in different formats and ineffective coordination between distributed different sensors and agents. Some of these problems were outlined by other professionals, but different approaches have been implemented to solve these problems. The proposed NNCDSS can be implemented in the multi-level agent framework to construct a robust, distributed, error tolerant and self protecting Distributed Intrusion Detection system.

References

[1] Loshin P., "Intrusion Detection", *Computer World,* <http://security.itworld.com/4363/CED010416STO59611/pfindex.html>, 2001

[2] Denning, D.E., An Intrusion Detection Model, *IEEE Transactions on Software Engineering*, vol. 13, p.p. 222-232, 1987.

[3] McHugh, J., Christie, A. & Allen, J., "Defending Yourself: The Role of Intrusion Detection Systems", *Software, IEEE,* Vol. 17, No. 5, p.p. 42-51, 2000.

[4] Amoroso, E & Kwapniewski, R, "A Selection Criteria for Intrusion Detection Systems," *Proc. 14th Ann. Computer Security Applications Conf.*, IEEE Computer Soc. Press, Los Alamitos, Calif., pp. 280-288, 1998.

[5] Burroughs, D. J., Wilson, L. F., & Cybenko, G. V. "Analysis of Distributed Intrusion Detection Systems using Bayesian Methods Performance," *IEEE Int'l Computing, and Communications Conference,* pp. 329 -334, 2002.

[6] Holz, T, 2004, "An efficient distributed intrusion detection scheme", *Computer Software and Applications Conference, COMPSAC Proceedings of the 28th Annual International,* Vol. 2, 28-30 Sept. 2004, pp. 39 - 40 *2004.*

[7] Kazienko, P. & Dorosz, P., *Intrusion Detection Systems (IDS) Part 2 - Classification; Method; Techniques*, WindowSecurity.com, <http://windowsecurity.com/pages/article_p.asp?id=1335>, 2004.

[8] Jirapummin, C.,Wattanapongsakorn, N. & Kanthamanon, P., Hybrid neural networks for intrusion detection system. *In Proceedings of The 2002 International Technical Conference On Circuits/Systems, Computers and Communications*, 2002.

[9] Cannady, J., Artificial neural networks for misuse detection. *In Proceedings of the National Information Systems Security Conference*, Arlington, VA, 1998.

[10] KDD dataset, <http://kdd..ics.uci.edu/databases/kddcup99/kddcup99.html>, 1999.

[11] Pfahringer B., Winning the KDD99 Classification Cup: Bagged Boosting. *SIGKDD Explorations*, Vol. 1, p.p. 65-66, 2000.

[12] Levin I., KDD-99 Classifier Learning Contest: LLSoft's Results Overview. *SIGKDD Explorations*, Vol. 1, p.p. 67-75, 2000.

[13] Miheev, V., Vopilov, A. & Shabalin, I., The MP13 Approach to the KDD'99 Classifier Learning Contest. *SIGKDD Explorations*, Vol. 1, p.p., 76-77, 2000.

[14] Zaknich, A., "Introduction to the modified probabilistic neural network for general signal processing applications", *IEEE Transactions on Signal Processing*, Vol. 46, No. 7, pp. 1980-1990, July 1998.

[15] Specht, D. F., "Probabilistic Neural Network", *Int. Journ. Neural Networks,* Vol. 3, pp. 109-118, 1990.

[16] Nadaraya, E. A., "On estimating regression", *Theory Probab. Appl.* pp. 141-142, 1964.

[17] Freund, Y. Schapire, R., "Experiments with a new boosting algorithm", *Proceedings of the Thirteenth International Conference on Machine Learning*, p.p. 148-156, Italy, 1996.

[18] Geman, S., Bienenstock, E., Doursat, R., Neural networks and the bias/variance dilemma, *Neural Computation*, vol. 4, p.p. 1-58, 1992.