©2010 International Journal of Computer Applications (0975 – 8887)
Volume 1 – No. 8

Self Organizing Maps to Build Intrusion Detection System

Mr. Vivek A. Patole
Dept. of Computer Engineering & Information Technology,
College of Engineering Pune,
Pune, India

Mr. V. K. Pachghare
Assistant Professor,
Dept. of Computer Engineering & Information Technology,
College of Engineering Pune,
Pune, India

Dr. Parag Kulkarni
Chief Scientist and Research Head
Capsilon Research Labs,
Capsilon India,
Pune, India

ABSTRACT

With the rapid expansion of computer usage and computer network the security of the computer system has became very important. Every day new kind of attacks are being faced by industries. Many methods have been proposed for the development of intrusion detection system using artificial intelligence technique. In this paper we will have a look at an algorithm based on neural networks that are suitable for Intrusion Detection Systems (IDS) [1] [2]. The name of this algorithm is "Self Organizing Maps" (SOM). Neural networks method is a promising technique which has been used in many classification problems. The neural network component will implement the neural approach, which is based on the assumption that each user is unique and leaves a unique footprint on a computer system when using it. If a user's footprint does not match his/her reference footprint based on normal system activities, the system administrator or security officer can be alerted to a possible security breach. At the end of the paper we will figure out the advantages and disadvantages of Self Organizing Maps and explain how it is useful for building an Intrusion Detection System.

Keywords: Neural Networks, Intrusion Detection System, Self Organizing Maps.

1. INTRODUCTION

Over the last few decades information is the most precious part of any organization. Most of the things what an organization does revolve around this important asset. Organizations are taking measures to safeguard this information from intruders. The rapid development and expansion of World Wide Web and local networks and their usage in any industry has changed the computing world by leaps and bounds [1][2].

INTRUSION DETECTION SYSTEM is a system that identifies, in real time, attacks on a network and takes corrective action to prevent them. They are the set of techniques that are used to detect suspicious activity both at network and host level. There are two main approaches to design an IDS.

1) MISUSE BASED IDS (SIGNATURE BASED)
2) ANOMALY BASED IDS.

In a misuse based intrusion detection system, intrusions are detected by looking for activities that correspond to know signatures of intrusions or vulnerabilities [3]. While an anomaly based intrusion detection system detect intrusions by searching for abnormal network traffic. The abnormal traffic pattern can be defined either as the violation of accepted thresholds for frequency of events in a connection or as a user's violation of the legitimate profile developed for normal behavior.

One of the most commonly used approaches in expert system based intrusion detection systems is rule-based analysis using Denning's profile model [3]. Rule-based analysis depends on sets of predefined rules that are provided by an administrator. Expert systems require frequent updates to remain current. This design approach usually results in an inflexible detection system that is unable to detect an attack if the sequence of events is slightly different from the predefined profile [4]. Considered that the intruder is an intelligent and flexible agent while the rule based IDSs obey fixed rules. This problem can be tackled by the application of soft computing techniques in IDSs. Soft computing is a general term for describing a set of optimization and processing techniques. The principal constituents of soft computing techniques are Fuzzy Logic (FL), Artificial Neural Networks (ANNs), Probabilistic Reasoning (PR), and Genetic Algorithms (GAs) [4].

SYSTEM ARCHITECTURE

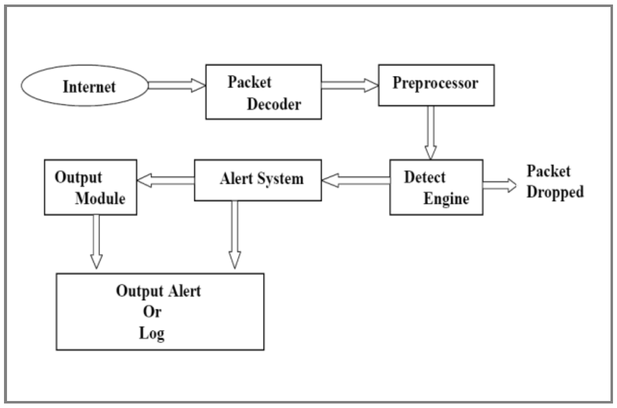

Figure 1. System Architecture.

2. TYPES OF NETWORKING ATTACKS

There are four major categories of networking attacks. Every attack on a network can be placed into one of these groupings [4].

- **Denial of Service (DoS):** A DoS attacks is a type of attack in which the hacker makes a memory resources too busy to serve legitimate networking requests and hence denying users access to a machine e.g. apache, smurf, Neptune, ping of death, back, mail bomb, UDP storm, etc.
- **Remote to User attacks (R2L):** A remote to user attack is an attack in which a user sends packets to a machine over the internet, and the user does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer, e.g. xlock, guest, xnsnoop, phf, sendmail dictionary etc.
- **User to Root Attacks (U2R):** These attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain super user privileges, e.g. perl, xterm.
- **Probing:** Probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system. This technique is commonly used in data mining, e.g. satan, saint, portsweep, mscan, nmap etc.

Two different attack types were included for this study: *SYN Flood (Neptune)* and *Satan*. These two attack types were selected from two different attack categories (denial of service and probing) to check for the ability of the intrusion detection system to identify attacks from different categories.

SYN Flood (Neptune) is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). For distinguishing a Neptune attack, network traffic is monitored for a number of simultaneous SYN packets destined for a particular machine. *Satan* is a probing intrusion, which automatically scans a network of computers to gather information or find known vulnerabilities. The purpose of classifiers in IDSs is to identify attacks from all four groups as accurately as possible.

3. SELF ORGANIZING MAPS (SOM)

The Self-Organizing Map is a neural network model for analyzing and visualizing high dimensional data. It belongs to the category of competitive learning network. The SOM Fig. 1 defines a mapping from high dimensional input data space onto a regular two-dimensional array of neurons. It is a competitive network where the goal is to transform an input data set of arbitrary dimension to a one- or two-dimensional topological map. The model was first described by the Finnish professor Teuvo Kohonen and is thus sometimes referred to as a Kohonen Map. The SOM aims to discover underlying structure, e.g. feature map, of the input data set by building a topology preserving map which describes neighborhood relations of the points in the data set [5].

The SOM is often used in the fields of data compression and pattern recognition. There are also some commercial intrusion detection products that use SOM to discover anomaly traffic in networks by classifying traffic into categories. The structure of the SOM is a single feed forward network, where each source node of the input layer is connected to all output neurons. The number of the input dimensions is usually higher than the output dimension.

The neurons of the Kohonen layer in the SOM are organized into a grid, see figure 2 and are in a space separate from the input space. The algorithm tries to find clusters such that two neighboring clusters in the grid have codebook vectors close to each other in the input space. Another way to look at this is that related data in the input data set are grouped in clusters in the grid [5]. The training utilizes competitive learning, meaning that neuron with weight vector that is most similar to the input vector is adjusted towards the input vector. The neuron is said to be the 'winning neuron' or the Best Matching Unit (BMU). The weights of the neurons close to the winning neuron are also adjusted but the magnitude of the change depends on the physical distance from the winning neuron and it is also decreased with the time.

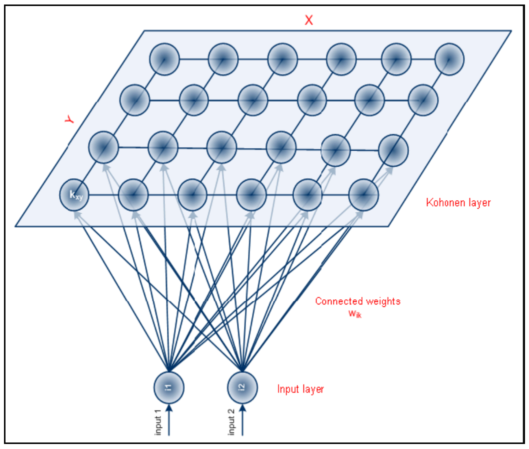

Figure 2. The self-organizing (Kohonen) map

The learning process of the SOM goes as follows:

1. One sample vector x is randomly drawn from the input data set and its similarity (distance) to the codebook vectors is computed by using Euclidean distance measure [6]:

$$\|x - m_c\| = \min_i\{\|x - m_i\|\} \quad \text{-------- (1)}$$

2. After the BMU has been found, the codebook vectors are updated. The BMU itself as well as its topological neighbors are moved closer to the input vector in the input space i.e. the input vector attracts them. The magnitude of the attraction is governed by the learning rate. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to the specified learning rate function type. Along with the learning rate, the neighborhood radius decreases as well. The update rule for the reference vector of unit i is the following:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x(t) - m_i(t)], i \in N_c(t) \\ m_i(t), i! \in N_c(t) \end{cases}$$

(2)

3. The steps 1 and 2 together constitute a single training step and they are repeated until the training ends. The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated accordingly.

After the training is over, the map should be topologically ordered. This means that *n* topologically close input data vectors map to *n* adjacent map neurons or even to the same single neuron.

4.1 Mapping Precision

The mapping precision measure describes how accurately the neurons respond to the given data set. If the reference vector of the BMU calculated for a given testing vector xi is exactly the same xi, the error in precision is then 0. Normally, the number of data vectors exceeds the number of neurons and the precision error is thus always different from 0. A common measure that calculates the precision of the mapping is the average quantization error over the entire data set [6]:

$$E_q = \frac{1}{N}\sum_{i=1}^{n} ||x_i + m_c||$$

(6)

4.2 Topology Preservation

The topology preservation measure describes how well the SOM preserves the topology of the studied data set. Unlike the mapping precision measure, it considers the structure of the map. For a strangely twisted map, the topographic error is big even if the mapping precision error is small. A simple method for calculating the topographic error [6]:

$$E_q = \frac{1}{N}\sum_{i=1}^{n} u(x_x)$$

(7)

Where $u(x_x)$ is 1 if the first and second BMUs of x_k are not next to each other. Otherwise $u(x_x)$ is 0.

4. EXPERIMENTS

5.1 Data Collection

If the network traffic has been examined carefully for different types of events such as downloading, port scanning, surfing etc., it is possible to identify the formal distinctions between them. The idea behind this work is to collect distinct and various kinds of network packets. To collect data we can use any packet sniffer which is available readily. Here in this case we have developed our own packet sniffer. Apart from capturing live packets we also a standard DARPA dataset, which we will be using for training purpose [7]. The dataset contain both packets with intrusion and without intrusion.

5.2 Vector Extraction

After the process of data collection, the features should be extracted from the data in order to obtain better classification results [8]. We slide time one by one, obtain new vectors and these obtained vectors will be used to provide input to self organizing map. Table 1. Shows examples of obtained vectors for time window length is equal to 5. We accepted window length as 20 for our application. Since the data are collected in every 20 seconds an input vector corresponds to time interval of 400 seconds

V1	V2	V3	V4	V5
16.1	14.3	24	125	128
14.6	25	125	133	11.2
23	122	134	10.5	16
122	129	10.3	18	19
132	10.6	16	19	15.5

Table 1. Table showing Extracted Vectors.

5.3 Training Self Organizing Map

For training purpose we constructed a 30x30 Self Organizing Map in order to perform clustering. The data that was used for it was DARPA dataset [7]. We used batch training algorithm with training length 100 and starting radius 15. Self organizing map was largely successful in classifying the IP packets.

5. RESULTS

After the data collection, vector extraction and training of the Self Organizing Maps we pass the packets through the SOM. The result is shown in the fig. The results Fig. 3 show input vectors classification, which represents behavior and its mapping to particular neurons, which form single possible user behavior states. Form states as intrusion – Intrusion, possible intrusion – Intrusion? Normal – Norm. From the test result SOM network represents suitable core for IDS systems [9] [10].

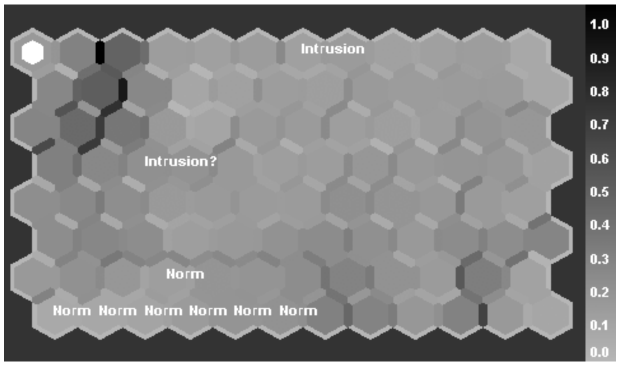

Figure 3. Results of the Experiment

6.1 Advantages and Disadvantages of SOM

6.1.1 Advantages:

1. Simple and easy-to-understand algorithm that works.
2. Topological clustering
3. Unsupervised algorithm that works with nonlinear data set.
4. The excellent capability to visualize high- dimensional data onto 1 or 2 dimensional space makes it unique especially for dimensionality reduction.

6.1.2 Disadvantages:

Time consuming algorithm, this is because as the no. of neurons affects the performance of the algorithm. And as the number increases the computation increases which results in increasing computational time [6] [7].

6. CONCLUSIONS

The Self Organizing Map is an extremely powerful mechanism for automatic mathematical characterization of acceptable system activity. In the above paper we have described how we can use Self Organizing Maps for building an Intrusion Detection System. We have explained the system architecture and the flow diagram for the SOM. We have also presented the pros and cons of the algorithm.

Our actual experiments show that even a simple map, when trained on normal data, will detect the anomalous features of both buffer overflow intrusions to which we exposed it. This approach is particularly powerful because the self organizing map never needs to be told what intrusive behavior looks like [11]. By learning to characterize normal behavior, it implicitly prepares itself to detect any aberrant network activity.

7. REFERENCES

[1] Damiano Bolzoni, Sandro Etalle, Pieter H. Hartel, and Emmanuele Zambon. Poseidon: a 2-tier anomaly-based network intrusion detection system. In Proceedings of the 4th IEEE International Workshop on Information Assurance, 13-14 April 2006, Egham, Surrey, UK, pages 144–156, 2006.

[2] D. A. Frincke, D. Tobin, J. C. McConnell, J. Marconi, and D. Polla. A framework for cooperative intrusion detection. In Proc. 21st NIST-NCSC National Information Systems Security Conference, pages 361–373, 1998.

[3] Denning D, "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, Vol. SE-13, No 2, Feb 1987.

[4] Simon Haykin, "Neural Networks: A Comprehensive Foundation", Prentice Hall, 2nd edition, 1999.

[5] Kohonen, T, "Self-Organizing Maps", Springer Series in Information Sciences. Berlin, Heidelberg: Springer. 1997.

[6] P. Lichodzijewski, A. Zincir-Heywood, and M. Heywood. "Dynamic intrusion detection using self organizing maps", 2002.

[7] McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. on Information and System Security 3 (2000) 262–294.

[8] Wenke Lee and Salvatore J. Stolfo, "A framework for constructing features and models for intrusion detection systems", ACM Trans. Inf. Syst. Secur., 3(4):227–261, 2000.

[9] Rhodes, B., Mahaffey, J., Cannady, J., "Multiple Self-Organizing Maps for Intrusion Systems"

[10] Bishop, C. M, "Neural Networks for Pattern Recognition", Oxford: Clarendon-Press, 1996.

[11] Lane, T., and Brodley, C. E. 1999. Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information and System Security 2(3):295—331.