Anomaly Detection in Ethernet Networks using Self Organizing Maps

Saroj Kumar Panigrahy, Jyoti Ranjan Mahapatra,
Jignyanshu Mohanty, and Sanjay Kumar Jena

Department of Computer Science and Engineering
National Institute of Technology Rourkela, 769 008, Odisha, India
{panigrahys,skjena}@nitrkl.ac.in

Abstract. Anomaly detection attempts to recognize abnormal behavior to detect intrusions. We have concentrated to design a prototype UNIX Anomaly Detection System. Neural Networks are tolerant of imprecise data and uncertain information. A tool has been devised for detecting such intrusions into the network. The tool uses the machine learning approaches ad clustering techniques like Self Organizing Map and compares it with the K-means approach. Our system is described for applying hierarchical unsupervised neural network to intrusion detection system.

Keywords: Intrusion detection, anomaly detection, self organizing map, neural networks.

1 Introduction

A secure computer network is one that assures data confidentiality, data and communications integrity and protection from denial of service (DOS) attacks [1]. The paradigm for securing computer networks was eventually replaced by the notion of intrusion detection [2]. There are many types of intrusion detection systems, but most can be classified in one of two ways [1]. First, an intrusion detection system can be classified based on the data source that it uses. A host-based intrusion detection system uses the audit trails of the operating system as a primary data source. For example, it may use records of user sessions to detect particular sessions that constitute an intrusion. A network-based intrusion detection system, on the other hand, uses network traffic information as its main data source. An example would be a system that uses TCP header information.

When analyzing any intrusion detection system, three factors must be considered—efficiency of the system, timeliness of detection and accuracy of detection [2]. The basic idea behind the system is that hierarchies of self organizing maps (SOM) take on a divide and conquer approach to concisely model the normal behavior of the system. Given the model of normal behavior, running data that corresponds to some suspicious behavior through the system will then exhibit some telltale signs that can be used to raise an alarm. The system described herein is a network based anomaly detection system that uses TCP dump dataset [3].

2 Advantages of SOM over *K*-means Approach

Authors are ambiguous over which is the best method for implementing the anomaly detection technique. Some support the SOM approach and some favour the K-means approach. But one major area of concern is not the difficulty of run time complexity but the difficulty of implementing it over dataset. The K-means approach needs a mean centroid to be defined at the outset of the run. But network traffic data is variable and deviates randomly over time. So, any randomly generated initial mean value is difficult to implement on variable network traffic. On the other hand SOM learns itself according to given data. Also k-means approach suffers from the problem of local optima. As the initial centroid value and number of clusters is chosen, it might be away from the optimal centroids and the end result for SOM is better than K-means. Search space is better explored by SOM due to the effect of the neighbourhood parameter which forces units to move according to each other in the early stages of the process. K-means gradient orientation forces a premature convergence which, depending on the initialization, may frequently yield local optimum solutions.

3 Implementation Details

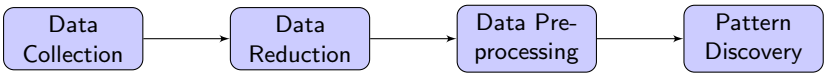
This section describes about the dataset, SOM architecture, data preprocessing, SOM training, calibrating The code book vectors, and Running the dataset referring with code book vector.

Dataset
The dataset available for constructing the system consisted of nearly five million connections of labeled training data and two million connections of test data. The connections were in chronological order. Each connection was described by 41 features. The features can be categorized as— *basic TCP features, content features, time based traffic features, and host based traffic features* [4]. A connection in the training data was either a normal connection or was one of 24 different attack types. Each connection was either normal or fell into one of the categories of attacks— *remote-to-local, user-to-root, denial-of-service, probing.*

From the available features, six were selected for use in the system— *duration, protocol type, service, flag, destination bytes, and source bytes.*Three of these features— *duration, destination bytes,* and *source bytes* had continuous values and *Protocol type, service,* and *flag* had discrete values. It should be noted that the entire dataset consisting of the seven million connections was not used in constructing the system. Only a 10% dataset from among the connection was used in order to make the training computationally feasible and most traffic has a typical pattern. Capturing the pattern of the traffic once is sufficient than doing it repeatedly. The 10% dataset represented the whole traffic connection for the training purpose. The dataset was extracted from the KDD Cup dataset which consisted of TCP dump data of DARPA Intrusion Detection Evaluation [3].

SOM Architecture

A SOM architecture with a single level was used [2]. Such an architecture was shown to be effective for the purpose of intrusion detection. The algorithm was fed with the above six parameters chosen. The data was preprocessed to get into a form that was program readable. To extract the TCP dump data, a network sniffer was used. The sniffer was placed on a central hub through which all traffic is routed so that it can capture all packets in promiscuous mode. It is a static dataset and used to standardize the algorithm for use on a more dynamic traffic data. The single level map model the behavior of the computer network with respect to time and given feature.The data flow in SOM is shown in following Figure.



Data Preprocessing

The first step in preprocessing the data involved removing all the attack connections from the training dataset, leaving only the normal ones. Care was taken to preserve chronological order. The second step in preprocessing the data involved extracting each feature from this file. This resulted in a sequence of feature values, one per feature. Next, because three of the six features consisted of discrete string values, a format that cannot be fed directly into the SOM, these features had to be enumerated. The result of the extraction and enumeration was six sequences of numbers, with each sequence corresponding to a feature. The nth entry in all of these sequences corresponded to the six features for the nth connection in the dataset. As is, if the values in each sequence were fed to the maps, no temporal relationship would have been encoded. In order to encode ordering and frequency relationships in the patterns that the maps would see, a FIFO buffer was used [5]. For a buffer of size n, the basic form of this algorithm is of following form:

1. The values of the sequence are fed into the buffer in chronological order.
2. Once the n positions of the buffer are filled, a pattern is generated.
3. When the next value in the sequence is observed, the oldest value currently in the buffer is discarded, the remaining values in the buffer shift by one so that the vacated position is filled and the next value is placed in the empty location. This generates the next pattern.

Training the SOM

The map was trained on a block of 15,000 consecutive connections, a fraction of the total dataset available after the first preprocessing stage. Although training on more patterns would allow the system to model a wider range of normal behavior, it would make the training of the maps difficult given the available time line. The maps were trained using C. The result was a 10×10 map. Training

uses all the mathematical calculations as described in [6]. For an input pattern given to the map, its distance to each mapping unit is found out and normalized. In this way, patterns close to map units yield a normalized distance close to one, and patterns far away from it yield a normalized distance close to zero. This normalization was done so that the values for all the features would have the same range. Otherwise, certain features would dominate the distance measure used in training, and thus the training of the map, simply because they had a larger range and not necessarily because they were more significant. For each pattern, the normalized distance to each center in the map was recorded, resulting in a six dimensional vector for each map. The vectors for all the maps were then concatenated to form one vector of dimension 100.

Calibrating the Code Book Vectors
The code book vectors were calibrated with carefully selected input patterns so that attack patterns are not there [6]. These normal input data patterns map to some of the mapping units and these mapping units are labeled as normal So the code book entries represent the anomalous as well as the normal patterns and are labeled. During testing any pattern that doesnt match to these units are termed anomalous and an alarm is raised.

Running the Dataset
The dump file from KDD dataset is run referring the code book vectors and the output is generated along with labels signifying which input patterns were termed as anomalous. The false positive and false negative rates are calculated based on the output type, whether it is anomalous or not, and the input pattern, whether it was actually an attack packet.

4 Results

The algorithm was run on the test dataset and according to the given architecture. First, the test data was preprocessed, the SOM was trained with training data, the code book entries were labeled by calibrating with normal connections, and then the resulting code book entries were used for running the algorithm. Using the SOM implementation, the following results were obtained.

- Dimension of grid used: 10×10
- Samples Taken for training: 15,000
- Samples taken for testing: 13,500
- Attacks detected: teardrop, portsweep, ipsweep, backdoor, nmap, neptune, satan, phf, warezmaster
- Attacks not detected: pod, buffer_overflow, guess_passwd, imap, ftp_write, toolkit
- Dataset: KDD 10% unlabelled training dataset and 10% labeled testing dataset
- False Positive rate: $2/13500 = 1.07\%$
- False Negative rate: $145/13500 = 0.015\%$

The efficiency of the anomaly detection tool is reflected from the false positive rate, false negative rate and the number of attacks that were detected. In this tool, false positive rate was found to be very low. On the other hand, in the small dataset taken, false negative rate was also found to be low. But the system is inefficient because a number of attacks were not detected— pod, buffer_overflow, guess_passwd, imap, ftp_write, toolkit. The low level of false negative rate was due to the reason that these are not DoS type of attacks and the six parameters we chose for implementing the algorithm were identical in all respects for these attack packets and the normal packets. As these packets are encountered less in number in the traffic, the numerator value in calculation becomes small, and hence the low false negative rate.

A major analysis would be around the detection of the mapping units of the 10×10 grid. The points of the grid where most normal packets match, i.e., the frequency with which the packets map to particular grid units. Figure 1 shows the number times each node in the top level map was the BMU for the normal training data. Clearly, some nodes are BMUs more frequently than others, but most of the nodes receive their fair share of hits. However, nodes 10, 17, 18, 26, 27, 28, 29, 38, 48, 55, 57, 58, 59, 60, 66, 67, 68, 69, 70, 79, 80, 89, 90, 99, 100 stand out because they receive relatively few hits. Thus, these nodes could be considered to be associated with abnormal behavior.

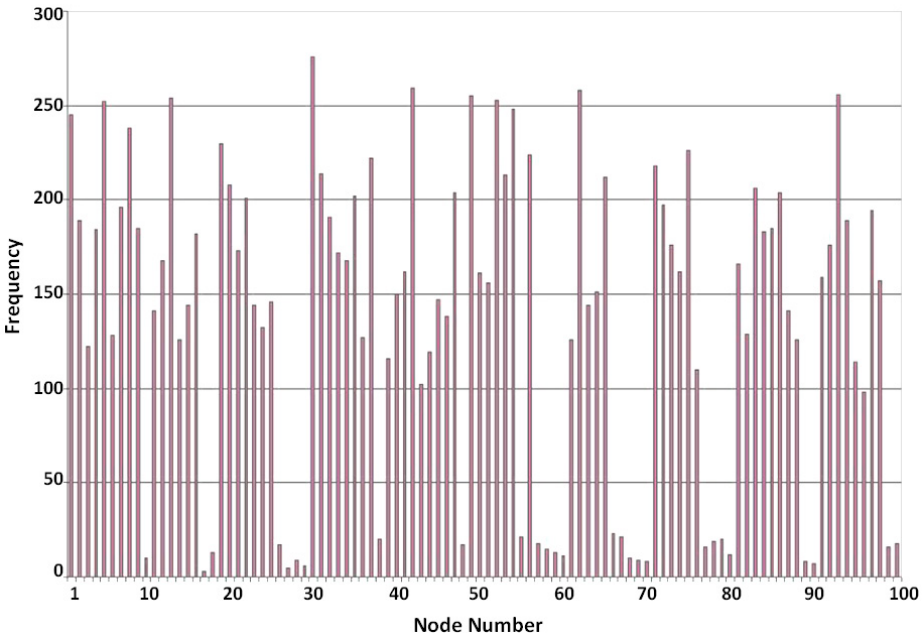


Fig. 1. Hits per node of normal training data

The performance of this system is comparable to that of the systems participating in the DARPA Intrusion Detection Evaluation 1999. The best system in the evaluation had an overall false negative rate of about 0.33 and an overall false positive rate of 0.0002. This system used all the available TCP connection features, and was trained on the entire available training data set which is shown in Table 1. Given that the system presented in this paper used only a fraction of this information, its performance is solid.

Table 1. Fraction of Data used

	Number of Connections Used	Fraction of Total
Training SOM	15,000	0.05
Labeling SOM	4,94,021	0.1
Testing SOM	3,11,029	0.1

5 Conclusions

A network based anomaly detection system that uses a hierarchy of SOMs was presented.The system was found to detect just over 60% of the attacks with a manageable rate of false alarms. Although the results of this work should be interpreted with caution, it is suggested that the system presented performs comparably to some of the better systems that took part in the DARPA Intrusion Detection Evaluation. The system was not tested on the full test dataset, i.e., it may not have encountered some of the more difficult attacks but it was also never trained on the full training dataset, meaning that it may not have had a chance to learn the full range of normal behavior.

References

1. Mukherjee, B., Heberlein, L.T., Levitt, K.N.: Network intrusion detection. IEEE Network 8(5), 26–41 (1994)
2. Lichodzjewski, P., Zincir-Heywood, A.N., Heywood, M.: Dynamic intrusion detection using self-organizing maps. In: Proceedings of the 14th Annual Canadian Information Technology Security Symposium. Ottawa, Canada (May 2002)
3. KDDCup: The third international knowledge discovery and data mining tools competition (May 2002), `http://kdd.ics.uci.edu/databases/kdd99cup/kdd99cup.html`
4. Lee, S.C., Heinbuch, D.V.: Training a neural network based intrusion detector to recognize novel attacks. IEEE Transactions on Systems, Man and Cybernetics (Jul 2001)
5. Lee, W., Stolfo, S.J., Chan, P.K., Eskin, E., Fan, W., Willer, M., Hershkp, S., Zhang, J.: Real time data mining based intrusion detection. In: Proceedings of DISCEX II (Jun 2001)
6. Kohonen, T.: Self Organizing Maps. Springer, 3rd edn. (2001)