Anomaly Detection in Ethernet networks Using Self Organizing Maps

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
In
Computer Science and Engineering

By

Jyoti Ranjan Mahapatra

Jignyanshu Mohanty

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2009

Anomaly Detection in Ethernet networks Using Self Organizing Maps

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
In
Computer Science and Engineering

By:

Jyoti Ranjan Mahapatra
Roll No.: 10506005

Jignyanshu Mohanty
Roll No.: 10506007

Under The Guidance of:

Prof. S.K.Jena

Department of Computer Science and Engineering
National Institute of Technology, Rourkela

May, 2009

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**ANAMOLY DETECTION IN ETHERNET NETWORK USING SELF ORGANIZING MAP**" submitted by Jyoti Ranjan Mahapatra and Jignyanshu Mohanty in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date:

Prof S.K.Jena
Dept. of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela - 769008

ACKNOWLEDGEMENT

We are sincerely thankful to **Prof S.K.Jena,** department of Computer Science and Engineering, NIT Rourkela for giving us the opportunity to work under him and lending every support possible at every stage of this project work. The level of flexibility offered by him in implementing the project work is highly applaudable. We would also like to thank **Prof. K.Sathya Babu** for extending his support during the entire duration of the project and giving us insights into the subject matter.

We would also like to convey our sincerest gratitude and indebtedness to all other faculty members and staff of the Department of Computer Science and Engineering, NIT Rourkela, who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on our part to finish the project work.

Date: **Jyoti Ranjan Mahapatra**

Date: **Jignyanshu Mohanty**

Abstract

The network is a highly vulnerable venture for any organization that needs to have a set of computers for their work and needs to communicate among them. Any large organization that sets up a network needs a basic Ethernet or wireless framework for transferring data. Nevertheless the security concern of the organization creeps in and the computers storing the highly sensitive data need to be safeguarded. The threat to the network comes from the internal network as well as the external network. The amount of monitoring data generated in computer networks is enormous. Tools are needed to ease the work of system operators. 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. We worked to devise a tool 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. The network connection is characterized by six parameters and specified as a six dimensional vectors. The self organizing map creates a two dimensional lattice of neurons for network for each network service. During real time analysis, network features are fed to the neural network approaches and a winner is selected by finding a neuron that is closest in distance to it. The network is then classified as an intrusion if the distance is more than a preset threshold. The evaluation of this approach will be based on data sets provided by the Defense Advanced Research Projects Agency (DARPA) IDS evaluation in 1999.

CONTENTS

List of Abbreviations

SOM	Self Organizing Maps
IDS	Intrusion Detection System
TCP	Transmission Control Protocol
DARPA	Defense Advanced Research Projects Agency
KDD	Knowledge Discovery and Data Mining
DOS	Denial of Service

List of Figures

Chapter

1

INTRODUCTION

1.1 Overview

A secure computer network is one that assures data confidentiality, data and communications integrity and protection from denial of service (DOS) attacks [1].Confidentiality deals with the need to guard one's private information, integrity mechanisms ensure that any change to any data must be done by authorized entities and through authorized mechanisms. Access control mechanisms are there that deal with protection against unauthorized access to data. A common approach to securing networks has been to control the flow of data in and out of the network. This approach, however was found to be impractical because it restricted user behavior, often required existing infrastructure to be discarded and was never really foolproof.

This paradigm for securing computer networks was eventually replaced by the notion of intrusion detection [2]. Any set of actions that threaten the confidentiality and integrity of the network resource is termed an intrusion in a general sense. Intrusion detection basically monitors and analyses events occurring in a computer or network system in order to detect signs of security problems. The basic steps in the working of intrusion detection system can be categorized as: monitoring and analyzing traffic, identifying abnormal activities, raising alarm. It was accepted that attacks could not be prevented altogether, but given that an attack is detected early enough, it should be possible to successfully defend against it. Thus the role of an intrusion detection system became to flag suspicious behavior for further analysis by a human. Although this approach is not itself foolproof, it does not suffer from many of the disadvantages of the old approach. Although it suffers from the problems of False Positives (an event, incorrectly identified as an intrusion when none has occurred) and False negatives (an event that IDS fails to identify as an intrusion when one has in fact occurred).

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.

An intrusion detection system can also be classified based on the intrusion detection technique that is used. There are two approaches in this regard: Signature based and Anomaly based. A rule or signature based system tries to identify behavior that is known to represent an attack. A signature is a set of rules pertaining to a typical intrusion activity. Since this technique looks for specific instances of abuse it is also known as misuse detection. Data mining techniques are used to record known intrusions in a database and referring to these patterns with incoming packets. An anomaly detection [2] system, on the other hand, tries to identify behavior that deviates from the normal [2]. A normal traffic pattern is first generated, threshold values are set for critical parameters chosen from among the network parameters. Whenever the traffic crosses any of these thresholds an alarm is raised. A problem with rule based systems is that they fail to identify novel attacks and may even miss known attacks if their signatures change [7]. Anomaly detection systems tend to suffer because normal behavior varies a lot and may change significantly over time, making the system less accurate.

When analyzing any intrusion detection system, three factors must be considered: efficiency of the system, timeliness of detection and accuracy of detection [10]. Host based and network based systems are not always efficient because the amount of available monitoring data is often overwhelming. Timeliness of intrusion detection is critical as it allows action to be taken against any threat that is detected. Accuracy of intrusion detection is also important because a system that raises too many false alarms is simply impractical. When there is a large amount of data to process, timeliness deteriorates. On the other hand, with less data, accurate decisions about the nature of the behavior may not always be possible.

The basic idea behind the system is that hierarchies of SOMs takes 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.

In order to analyze and evaluate the current intrusion detection systems, MIT Lincoln Labs organized the DARPA 1998 Intrusion Detection Evaluation. Provided was TCP dump data generated over nine weeks by a simulated military local area network. It contained normal traffic as well as attacks. This data was Processed into some seven million TCP connection records for use in The Third International Knowledge Discovery and Data Mining Tools Competition held in 1999 [4]. The system described herein is a network based anomaly detection system that uses this latter dataset. The rest of the paper is organized as follows. Section 2 introduces the basis for the system, the SOM. The process used in constructing the system is described in section 3. Section 4 discusses the results of testing the system, section 5 discusses future work, and section 6 states the conclusions that can be derived from this work.

1.2 Motivation

The major impetus for carrying out this project work and entering this area has been our summer internship at Glorytech, Bhubaneswar. The organization deals with the networking needs of the local vendors and we got an insight into how networks are organized. We got to work on static firewalls to safeguard the networks. We studied AI and machine learning in 6th semester and that led us to think about implementing clustering approaches to network traffic data to make a anomaly detection tool that would use machine learning to configure itself and observe patterns among network traffic. This area has been taken up by researchers and various means suggested to do the implementation. We took the SOM architecture to implement our tool because of its efficiency.

1.3 Objectives

- To build a neural network model to implement Self Organizing Map.
- To implement a means to extract network parameters from network traffic.
- To provide a way to implement the SOM algorithm on the extracted features.
- To provide a way to calibrate the results with normal traffic and identify attacks as anomalous.
- Compare with K-means approach.

Chapter

2

CLUSTERING APPROACHES

Myriads of clustering approaches have been devised by researchers in order to implement machine learning algorithms. Our work concentrated on the study of the Self Organizing Maps (SOM) and the K-means approach. These two approaches were extensively studied and compared. This section deals with the basics of the two approaches and their advantages and disadvantages over each other. The reason of using SOM over K-means is described.

2.1 Self Organizing Maps (SOM)

The concept, design, and implementation techniques of Self-Organizing Maps are described in detail in [11]. The algorithm converts non-linear statistical relationships between data points in a high-dimensional space into geometrical relationships between points in a two-dimensional map, called the Self-Organizing Map (SOM). A SOM can then be used to visualize the abstractions (clustering) of data points in the input space. The points in the SOM are called neurons, and are represented as multidimensional vectors. If the data points in the input space are characterized using k parameters and represented by k-dimensional vectors, the neurons in the SOM are also specified as k-dimensional vectors. The SOM is an unsupervised neural network algorithm that uses competitive learning [6]. Competitive learning means that as data is input to the SOM, there is a competition among the neurons or nodes of the map to determine which neurons will represent the input data. In the case of the SOM, the winning neuron is the neuron most similar to the input data, and it is affected by becoming more like the input data. In this way, neurons in the map become specialized to represent different sets of data in the input space.

The SOM consists of a two dimensional grid of neurons. Each neuron is represented by a prototype vector of weights whose dimension is equal to that of the input data. Associated with each neuron is a neighborhood relation. This relation dictates the map's topology, or who the neighbors of a neuron are. Based on the topology, distance in the map can be defined. For example, the nodes adjacent (directly connected) to a given neuron are within a radius of one from that neuron.
One important property of the SOM is that it provides a topology preserving mapping from the input space to the two dimensional grid of nodes [5]. This means that points that are close in the input space are mapped to units that are close in the output space.
The steps of the SOM follow.

2.1.1 Learning

In the SOM Learning phase, the neurons in the SOM are trained to model the input space. This phase has the following two important characteristics:

– Competitive. Each sample data point from the input data space is shown in parallel to all the neurons in the SOM, and the “winner” is chosen to be the neuron that responds best. The k-dimensional values of the winner are adjusted so that it responds even better to similar input.

– Cooperative. A neighborhood is defined for the winner to include all neurons in its near vicinity in the SOM. The k-dimensional values of neurons in the neighborhood are also adjusted so that they too respond better to a similar input.

2.1.2 Distance Measure

For the purpose of locating the winner neuron given the data sample, a suitable measure of distance has to be defined. The commonly used distance measures are the Euclidean and the Dot-product measures. In the Euclidean measure, given two points X ($x_1, x_2, . . ., x_k$) and Y ($y_1, y_2, . . ., y_k$) in k-dimensional space, the Euclidean distance is given by

$$\sqrt{}((x_1-y_1)^2 + (x_2-y_2)^2+..+(x_k-y_k)^2)$$

If the Dot-product measure is to be used, the input data points and the neurons in the SOM have to be normalized. Normalization of a vector V ($v_1, v_2, . . .,v_k$) is a process of transforming its components into

$$(v_1/\sqrt{}\ v_1^2+ v_2^2+..+ v_k^2\ ,\ v_1/\sqrt{}\ v_1^2+ v_2^2+..+ v_k^{2,\dots}\ ,\ v_1/\sqrt{}\ v_1^2+ v_2^2+..+ v_k^2)$$

so that the modulus of the normalized vector is unity. The dot-product of the input data point is calculated individually with each of the neurons, where the dot-product of two normalized vectors X ($x_1, x_2, . . ., x_k$) and Y ($y_1, y_2, . . ., y_k$) is defined to be

$$x_1.y_1 + x_2.y_2 + . . . + x_k.y_k$$

The winner is selected to be the neuron that gives the maximum dot product.

Training of the map is done by feeding individual patterns to the map [3]. When a training pattern x is shown to map, it is compared the all the prototype vectors in the map using some distance measure d. This is often just the Euclidean distance. The best matching unit BMU is then defined to be a prototype vector m_c such that

$$d(x,m_c) = \min_i d(x,m_i)$$

where i iterates over all the nodes in the map. Once the BMU is found, each node m_i in the map is updated via the update rule

$$m_i(t+1) = m_i(t) + h_{ci}(t) [x - m_i(t)]$$

where *mi*(*t*), *mi*(*t*+1) represent k-dimensional values of neuron *i*, at time *t* and *t* + 1 respectively; *x*(*t*) represents the k-dimensional values of the sample data(input pattern), t is the point in time, and h_{ci} a time-variable non-increasing neighborhood function about m_c.

2.1.3 Neighborhood Function

The commonly used neighborhood functions are Gaussian and Bubble. In the Bubble function, the neighborhood radius is specified by a variable *σ*, and all neurons within the neighborhood are adjusted by the same factor *α* towards the winner. The parameter *α*, called the learning rate factor, and the neighborhood size *σ*, are generally chosen to be monotonically decreasing functions of time *t*, where *t* is a discrete time measure incremented with every iteration of the training process.
The Bubble neighborhood function $h_{ci}(t)$ is specified as:

$$h_{ci}(t) = \begin{cases} \alpha(t) & , \|rc, ri\| < \sigma(t) \\ 0 & , \text{otherwise} \end{cases}$$

The Gaussian neighborhood function adjusts the winner neuron the most towards the sample data, and adjusts the remaining neurons within the neighborhood lesser and lesser as their distance from the winner increases, based on a bell shaped Gaussian function. It is specified as:

$$h_{ci}(t) = \begin{cases} \alpha(t) \exp(-\|rc,ri\|^2 / 2\sigma^2(t)) & , \|rc, ri\| < \sigma(t) \\ 0 & , \text{otherwise} \end{cases}$$

Here α is the learning rate. It is set to a high value early on in the training to produce a rough training phase and then decreases with time to fine-tune the training. The function $exp\ (-||r_c, r_i||^2 / 2\sigma^2\ (t))$ serves the purpose by defining the radius about m_c beyond which nodes in the map are not significantly affected by the input pattern x. This radius also tends to decrease over time. Figure 1 shows the basic SOM training algorithm. The map shown is a 3×4 map with a one-dimensional input space. A pattern with scalar value 6.4 is fed to the map. It is compared to all the nodes in the map using the Euclidean distance. and is found to be most similar to the node weighed 6. The example shows the case where the winning node and its immediate neighbors (shown in bold) are updated to reflect the input pattern more closely.

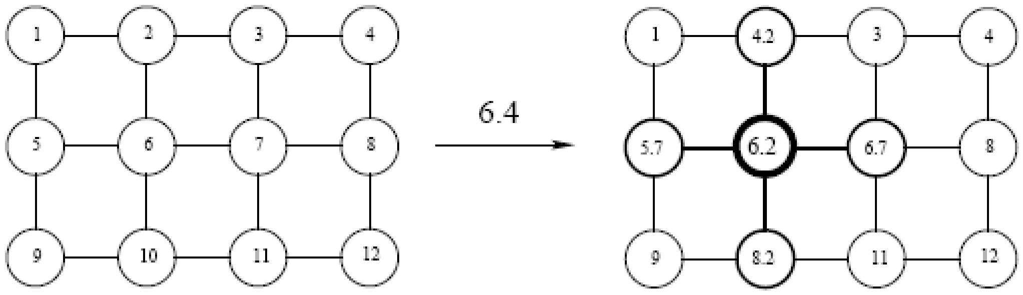


Figure 1: SOM training algorithm

To summarize, for every neuron in the SOM, the learning function calculates
its distance from the sample data, and adjusts its k-dimensional values towards
the sample data by a factor specified by the neighborhood function $h_{ci}\ (t)$.
Since the SOM is often used on high dimensional data, it is often difficult to visualize exactly what a map may look like. Several projection techniques have been developed. These aim to reduce the dimension of the vectors in the map while maintaining their relationships. Another way of visualizing an SOM is through a united matrix, a representation that shows distances between adjacent nodes using color.

The SOM is a useful analysis and visualization tool. It is capable of representing the features of an underlying dataset in a concise manner. In the context of an intrusion detection system, the SOM can be used as a concise model of the normal behavior on the network.

2.2 K-Means Clustering

Yet another method to analyze anomalous and normal traffic data is the K-means approach. The resulting cluster centroids are then used for fast anomaly detection in monitoring data. K-means clustering [12] is a clustering analysis algorithm that groups objects based on their feature values into K disjoint clusters. Objects that are classified into the same cluster have similar feature values. K is a positive integer number specifying the number of clusters, and has to be given in advance. Here are the four steps of the K-means clustering algorithm:

- Define the number of clusters K.
- Initialize the k cluster centroids. This can be done by arbitrarily dividing all objects into k-clusters, computing their centroids and verifying that all centroids are different from each other. Alternatively the centroids can be initialized to K arbitrarily chosen, different objects.
- Iterate over the objects and compute the distances to the centroids of all clusters. Assign each object to the cluster with the nearest centroid.
- Recalculate the centroids of both modified clusters taking learning rate into consideration.
- Repeat step 3 until the centroids do not change any more.

A distance function is required in order to compute the distance (i.e. similarity) between two objects. The most commonly used distance function is the Euclidean one which is defined as:

$$d(x, y) = \sqrt{\sum (xi - yi)^2}$$

where $x = (x_1, \ldots., x_m)$ and $y = (y_1 \ldots y_m)$ are two input vectors with m quantitative features. In the Euclidean distance function, all features contribute equally to the function value. However, since different features are usually measured with different metrics or at different scales, they must be normalized before applying the distance function. The recalculation of centroids is based on the formula:

$$m_{new} = m_{old} + \eta(x_t - m_{old})$$

where η is a small positive learning rate.
An alternative to Euclidean distance is the Mahalanobis distance function that uses the inverse covariance matrix S^{-1} to reflect statistical correlations between different features:

$$d(x, y) = q(x - y)^T S^{-1} (x - y)$$

However, calculating and inverting the covariance matrix is computationally demanding for feature vectors with a large number of dimensions.
K-means clustering algorithm can be applied to training datasets which may contain normal and anomalous traffic without being labeled as such in advance. The rationale behind this approach is the assumption that normal and anomalous traffic form different clusters in the features space. Of course, the data may contain outliers which do not belong to a bigger cluster, yet this does not disturb the K-means clustering process as long as the number of outliers is small. As already mentioned, the clustering is done individually for the predefined services,
identified by their typical *(protocol, port)* pair, as well as for the default classes that cover the remaining flows distinguished by the *protocol* value only.

The clustering algorithm divides the training data into K clusters, but does not determine if a cluster reflect time intervals of normal or anomalous traffic. This decision has to be made manually or by heuristics. For example, a higher
average in the number of packets can be taken as an indicator for an anomalous cluster.
It may occur that clusters are very close to each other. This can have several reasons: Either the number of clusters K has been badly chosen or the training data is very homogeneous, e.g. because it does not contain any anomalous traffic or because the anomalous traffic looks very similar to normal traffic. Nevertheless, the cluster centroids can still be used for outlier detection.
An essential problem of the K-means clustering method is to define an appropriate number of clusters K. As initial value, we chose K = 2, assuming that normal and

anomalous traffic in the training data form two different clusters. Obviously, a different number of clusters may result in better clusters, e.g. if the considered service already shows distinct periods of very low and very high traffic volume under normal conditions.

Classification and Outlier Detection

The K-means clustering process results in cluster centroids for normal and anomalous traffic which can be used to detect anomalies in new flow records monitored in the same network. New flow records have to be preprocessed and transformed like the training data in order to obtain the same features. For the purpose of anomaly detection, there are two distance based methods -classification and outlier detection, that both use the K-means clustering results and that can be applied individually or in a combined way.

Classification

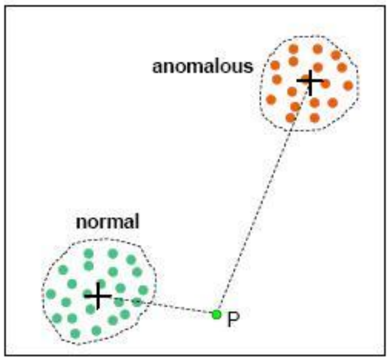
The distances to the cluster centroids of the corresponding traffic class are calculated using the weighted Euclidean distance function. An object is classified as normal if it is closer to the normal cluster centroid than to the anomalous one, and vice versa. This is illustrated in Figure 2(b) with a two-dimensional feature space: Object P is closer to the normal cluster, therefore P is normal. This distance-based classification allows detecting known kinds of anomalies, i.e. anomalous traffic with similar characteristics as in the training datasets.

Outlier detection

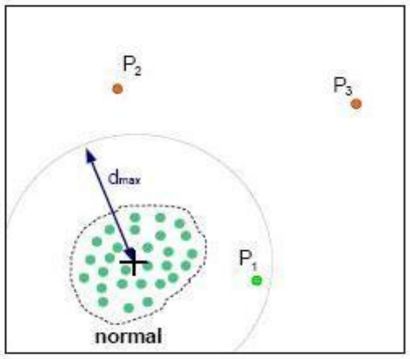
An outlier is an object that differs from most other objects significantly. Therefore it can be considered as an anomaly. For outlier detection, only the distance to the appropriate centroid of the normal cluster is calculated. If the distance between an object and the centroid is larger than a predefined threshold d_{max}, the object is treated as an outlier and anomaly. This is depicted in Figure 2(c) where P2 and P3 lie outside the d_{max} circle. In contrast to the classification method, outlier detection does not make use of the anomalous cluster centroid, i.e. it may be less accurate in detecting known kinds of anomalies. On the other hand, it allows detecting new anomalies that do not appear in the training datasets.

Combined classification and outlier detection

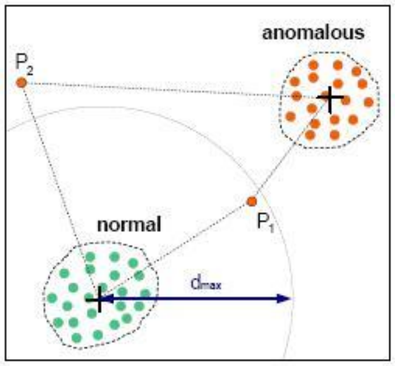
Classification and outlier detection can be used in a combined way in order to overcome the limitations of each individual method. If the two methods are applied simultaneously, an object is treated as an anomaly if it is closer to the anomalous cluster centroid than to the normal one, or if its distance to the normal cluster centroid is larger than the predefined threshold. In figure 4, for example, both objects P1 and P2 are regarded as anomalies. P1 is closer to the anomalous cluster and P2's distance to the normal group is larger than the threshold $d_{max.}$



(a)



(b)



(c)

Figure 2: K-means algorithm (a) Finding nearest neighbor (b) Finding dmax (c) Labeling P1 normal and P2 anomalous, based on euclidean distance

2.3 Reasons for using SOM implementation 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 favor 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. This is due to the effect of the neighborhood 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.

Chapter

3

IMPLEMENTATION

3.1 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 follows [7]

Basic TCP features: These features include the duration, protocol type, and service of the connection, as well as the amount of data transferred.

Content features: These features were derived from the payload of the TCP packets using domain knowledge. They include features like the number of failed login attempts and whether or not root access was obtained.

Time based traffic features: Calculated over a two second time interval, these features include things like the number of connections to the same host as the current connection and the number of connections to the same service as the current connection.

Host based traffic features: Analogous to the time based traffic features, host based traffic features are derived over the past 100 connections. They are meant to catch attacks that span longer than two seconds.

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 following categories of attacks.

Remote-to-Local: The intruder attempts to gain unauthorized access from a remote machine. For example, this may involve guessing a password.

User-to-Root: The intruder tries to access the superuser account by using for example, some form of buffer overflow.
Denial-of-Service: The intruder attempts to reduce the performance of a host, possibly going as far as making the host unavailable.

Probing: The intruder attempts to gather information about the host.

From the available features, six were selected for use in the system. All six were basic TCP features. There were two main reasons for this. First, selecting more features would have made the task of training the system computationally infeasible given the available time. Second, there was interest in determining just how far an entirely data driven machine learning paradigm could be pushed, and most of the other features were based on a priori knowledge [9]. Also, with the pure data driven approach, it is possible that some of the other features and the relations between them may in fact be learned by the system.

The selected features were:

Duration: The length (in seconds) of the connection.
Protocol type: The protocol of the connection, such as TCP or UDP.
Service: The service accessed by the connection, such as HTTP or Telnet.
Flag: The status flag of the connection.
Destination bytes: The amount of data sent by the destination of the connection.
Source bytes: The amount of data sent by the source of the connection.

Three of these features: duration, destination bytes, and source bytes, had continuous values. Protocol type, service, and flag all 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 tcpdump data of DARPA Intrusion Detection Evaluation [4].This represents TCP dump data generated over nine weeks of simulated network traffic in a hypothetical military local area network. This data was processed into some 7 million TCP connection records for use in the 3rd International Knowledge Discovery and Data Mining Tools Competition in 1999 [4].

3.2 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 six parameters chosen, connection duration, protocol type, service, flag, destination byte, source byte.

The data was preprocessed to get into a form that was program readable. The extract the tcpdump 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. Using this program on a Personal computer would lead to suboptimal results as it has slow network adapter and low buffer space. This sniffer can only be used on a dedicated server. For our purpose the standard dataset of DARPA IDS was used. It is a static dataset and used to standardize the algorithm for use on a more dynamic traffic data.

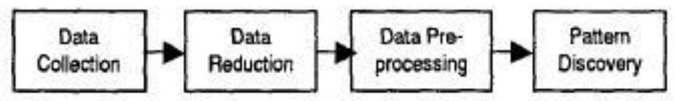


Figure 3: Data Flow in SOM

The single level map model the behavior of the computer network with respect time and given feature. Here, time does not refer to explicit points in time when values are observed to occur, but rather to the relative ordering and frequency of the values. Based on the output of the map, the network administrator should be able to decide whether or not a particular connection is an attack.

3.3 Data Preprocessing For Training the SOM

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 conjecture was that by training the maps on patterns that corresponded to normal network behavior, the hierarchy of maps would model this normal behavior. Within each map, the majority of the prototype vectors would be similar to patterns that correspond to normal behavior on the network. These would fall within dense regions of the map. However, as the map expands and contracts to represent the input data, it would have to stretch across regions that do not correspond to normal behavior and may isolate regions. These would then be sparse regions in the map. If a pattern would excite a node in one of these latter regions, it could then be considered abnormal and tagged as an attack.

The original dataset consisted of a comma delimited file where each line in the file corresponded to one connection. 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. For example, when the duration feature was extracted, the result was a sequence of durations.

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. Basically, this involved iterating over the sequence, replacing each discrete value by an integer. If a discrete value did not map to any integer (i.e. was not seen before), it was mapped to the next lowest available integer.

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 first-in-first-out FIFO buffer was used [8]. For a buffer of size n, the basic form of this algorithm takes the 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.

This algorithm was applied to each sequence. In this way, temporal information, be it only the relative order of values, was implicitly encoded in the patterns. This is important because often an attack may not be identified based solely on the features of one TCP connection, but on that of several, successive, TCP connections. For example, a DOS attack may consist of several successive connections with a specific value for a particular feature. In addition, features similar to the host based traffic features may become visible to the map because the information that the map sees spans many connections.

The result of this preprocessing stage was six sets of twenty dimensional patterns, one for each feature, generated from normal connections.

3.4 Training The SOM

The map was trained on a block of 15000 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 timeline. The maps were trained using C . The result was a 10×10 map. Training uses all the mathematical calculations as described in chapter 2. For an input pattern given to the map, its distance to each mapping unit is found out. This distance was then 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.

3.5 Calibrating The Code book vectors

The Code book vectors were calibrated with carefully selected input patterns so that attack patterns are not there. 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 doesn't match to these units are termed anomalous and an alarm is raised.

3.6 Running the dataset referring with code book vector

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's anomalous or not, and the input pattern, whether it was actually an attack packet.

3.7 System Architecture Overview

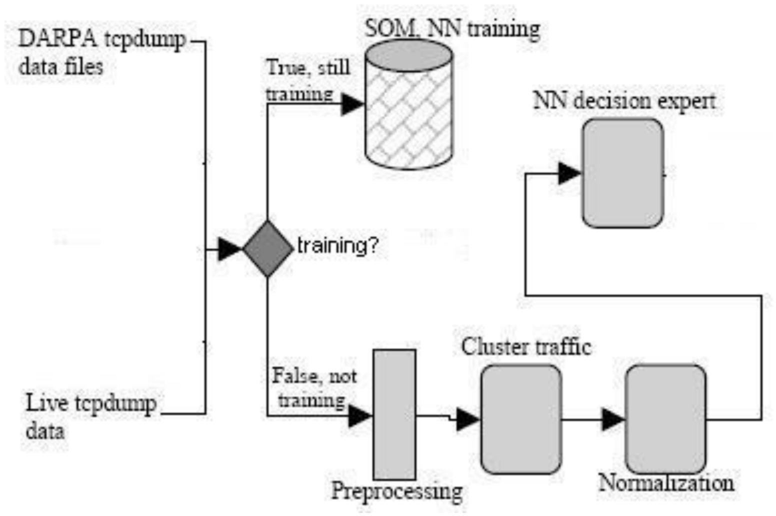


Figure 4: Proposed Architecture

Chapter 4

RESULTS

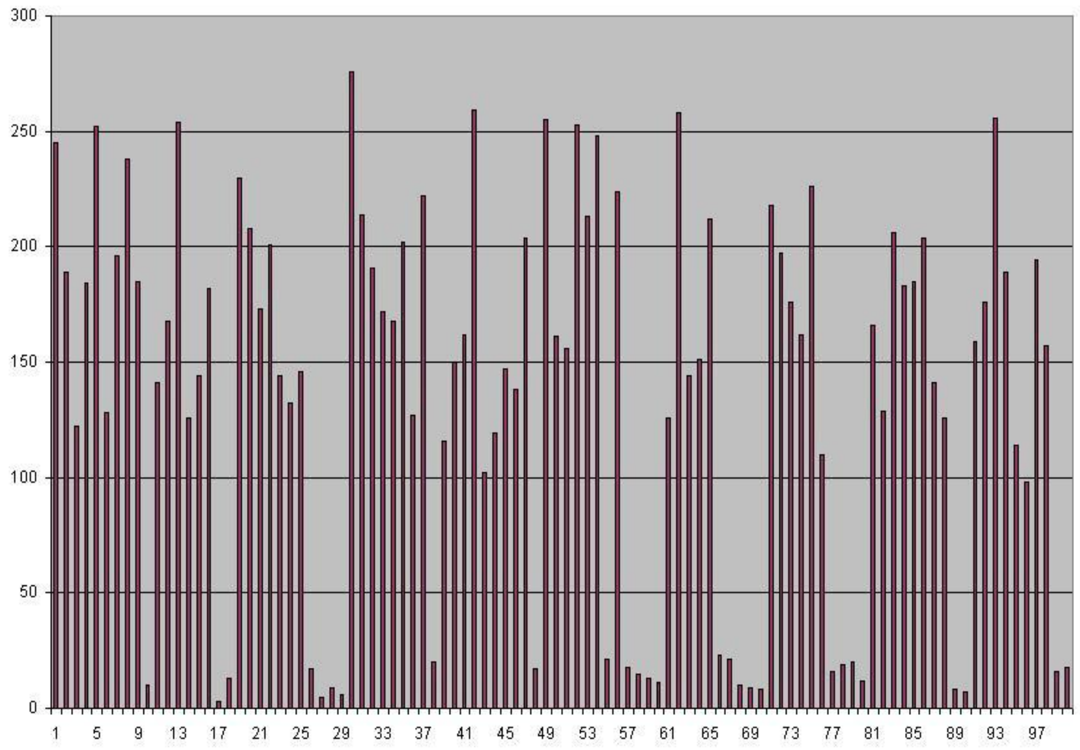
The algorithm was run on the test dataset and according to the proposed architecture. First the test data was preprocessed, the SOM was trained with training data, the code book entries were labeled by calibrating with normal connections and then the resulting code book entries were used for running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

- Dimension of grid used : 10*10
- Samples Taken for training : 15000
- 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 denial of service 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 (in other words the frequency with which the packets map to particular grid units). Figure shows how many times each node in the top level map was the BMU for the normal training data (the 15000 patterns used to train the map). Clearly, some nodes are BMUs more frequently than others, but most 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.

frequency



Node number

Figure 5: Hits per node of normal training data

One rule for labeling a connection as an attack might then be to check these BMUs, and if they are all identified nodes, label that connection as an attack. Otherwise, the connection would be considered normal. Testing of the system was performed on roughly 0.1% of the total available test dataset without regard to the attack types.

The number of false positives and false negatives was calculated. A false positive was defined to be a normal connection that was identified as an attack. A false negative was defined to be an attack connection that was not detected. Thus, false positives measure the false alarm rate while false negatives measure the detection rate. Using the rule stated above, the false positive rate was found to be about 1.07 % while the false negative rate was roughly 0.015 %. A false positive rate above one percent presents a problem because then the network administrator is forced to check the over one in a hundred connections that are tagged. On a busy network, this number becomes unmanageable rendering the system useless. The false positive rate of 1.07 % can be attributed to the fact that anomalous nodes are BMUs frequently in the normal data as well.

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. Given that the system presented in this paper used only a fraction of this information its performance is solid.

	Number of Connections Used	Fraction of Total
Training SOM	15000	0.05
Labeling SOM	494021	0.1
Testing SOM	311029	0.1

Chapter

5

FUTURE WORK

The project presented in this paper opens up many avenues for future work. For example, many decisions made during the development of the system were made with little deliberation due to time constraints. Should more resources be available, the following issues could be revisited.

- The current system was based on six basic TCP connection features. The result of adding more features or changing the features that are used could be explored in more detail.
- Only about 0.5 % of the total available training dataset was used. The system should be trained on the full training dataset.
- The FIFO buffer algorithm is an important part of the system. Experimentation could be performed on the size and sampling rate of the algorithm.
- Testing of the system was done on about 10 % of the test dataset. Testing of the system should be based on the full test dataset, and with regard for the different attack types.
- There are different sequences of BMUs that could define when an attack situation is taking place with each definition resulting in different performance. The definitions that were examined were only heuristics and were fairly arbitrary. Other such rules could be explored to see if they catch more attack cases while raising fewer false alarms.

The system could also be expanded in order to determine if its performance could be improved. For example, another level could be added to the existing SOM architecture. This level would only see patterns that were considered suspicious by the current single level map. Since this new level would see only a fraction of the connections that the lower levels see, it may be able to make a better distinction between normal and attack patterns compared to the heuristic used in obtaining the results. The additional level in the architecture would also make the manual analysis of the results and definition of attack situations based on BMU sequences unnecessary.

An unsupervised anomaly detection system is one that does not require any data to be labeled [6]. The benefit of such a system is that it does not require the labeling of the training dataset for each deployment of the system into a new environment. As is although it uses an unsupervised machine learning technique, the current system is not an unsupervised anomaly detection system in the sense because labels were needed to partition the training dataset. Another area of future work would therefore be to transform the system into a genuine unsupervised anomaly detection system.

Chapter

6

CONCLUSION

A network based anomaly detection system that uses a hierarchy of SOMs was presented. The dataset used in the system was derived from that used in the DARPA Intrusion Detection Evaluation. The hierarchy consisted of one level. Once the description of the system was given, rules for identifying intrusions were presented. Using these rules 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, meaning that 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.

In addition the system was for the most part entirely data driven. The work presented in this paper is still preliminary and there are many areas where the outlined system could be improved upon. However it has been worthwhile because it not only validates the results of previous work on SOMs and intrusion detection but it suggests that the approach may eventually lead to a system that provides timely and accurate results with a high level of efficiency. Although the work did not produce an intrusion detection system that could be successfully.

Deployed on real networks a promising start towards that goal was made. The fact that there are so many areas in the system that could be further investigated is also encouraging because it means that there is plenty of room for improvement.

References

[1]Biswanath Mukherjee, L. Todd Heberlein and Karl N. Levitt. Network intrusion detection.IEEE Network 8(5):26-41,May 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, 13-17 May 2002)

[3] S.L.Chiu.Fuzzy model identication based on cluster estimation. *Journal of Intelligent and Fuzzy Systems .* 2(3) page:267:278,1994.

[4]. The Third International Knowledge Discovery and Data Mining Tools Competition. Http://kdd.ics.uci.edu/databases/kdd99cup/kdd99cup.html. May, 2002

[5] Jaakko Hollmen Process modeling using the self organizing map Master's thesis, Helsinki University of Technology .

[6] Samuel Kaski Data exploration using self organizing maps. *Acta Polytechnica Scandinavica Mathematics Computing and Management in Engineering Series No .82 ,March 1997*

[7] Susan C.Lee and David.V.Heinbuch. Training a neural network based intrusion detector to recognize novel attacks. *IEEE Transactions on Systems, Man and Cybernetics,*July 2001.

[8] Wenke Lee, Salvatore J.Stolfo, Philip K Chan, Eleazar Eskin, Wei Fan, Matthew Willer,Shlomo Hershkp and Junxin Zhang. Real time data mining based intrusion detection. Proceedings of DISCEX II ,June 2001.

[9] Wenke Lee, Salvatore J.Stolfo and Kui W. Mok. Mining in a dataflow environment: Experience in network intrusion detection In Knowledge Discovery and Data Mining, pages 114-124,1999.

[10] Peter Lichodzijewski, A. Nur Zincir- Heywood and Malcolm I.Heywood. Host based intrusion detection using self organizing maps. *In proceedings of the IEEE World Congress on Computational Intelligence ,2002*

[11] T.Kohonen. *Self Organizing Maps*. Springer, third edition, 2001.

[12] Traffic Anomaly Detection Using K-Means Clustering, Gerhard M¨unz, Sa Li, Georg Carle, Computers and Communications, 2008. ISCC 2008. IEEE Symposium