



Reducing IDS false positives using Incremental Stream Clustering (ISC) Algorithm

Master's Thesis Report

Champa Dey

**Dept of Computer and Systems Sciences,
Royal Institute of Technology, Sweden.**

13th March, 2009

This thesis corresponds to 20 weeks of full-time work

TABLE OF CONTENTS

<i>Abstract</i>	VI
1 Introduction	1
1.1 Background	1
1.2 Problem	2
1.3 State of art.....	2
1.4 Hypothesis.....	4
1.5 Goal	4
1.6 Purpose	4
1.7 Research Methodology	5
1.8 Limitations	6
1.9 Intended Audience.....	6
1.10 Thesis Outline.....	6
2 Extended Background	7
2.1 Brief Details about IDS.....	7
2.2 Machine Learning.....	9
2.2.1 Supervised Learning Algorithm.....	9
2.2.2 Unsupervised Learning Algorithm.....	9
2.2.3 Reinforcement Learning.....	10
2.3 Data Mining.....	10
2.4 Machine Learning in Intrusion Detection.....	10
2.4.1 Classification.....	10
2.4.2 Clustering.....	11
2.4.3 Artificial Neural Network.....	11
2.4.4 Bayesian Classification.....	11
3 Our Proposed IDS False Alarm Filter with ISC Algorithm	12
3.1 Incremental Stream Clustering (ISC) Algorithm.....	12
3.1.1 Training phase of ISC algorithm.....	12
3.1.2 Testing phase or anomaly detection phase of ISC Algorithm.....	14
3.1.2.1 The First Approach.....	14
3.1.2.2 The Second Approach.....	14
3.2 Running Time of the ISC Algorithm.....	15
3.3 Incremental Stream Clustering Model for False Alert Reduction:	15
3.4 Implementation of the ISC Algorithm.....	16

4 Testing and Verification of the ISC Algorithm.....	17
4.1 Data Source.....	17
4.2 Experimental Setup.....	18
4.3 Testing with DARPA 1999 Dataset.....	23
4.4 Measurements for Experimental Validation.....	25
4.5 Results.....	27
4.6 Strength of the ISC Algorithm against KNN Algorithm for filtering IDS alarms.....	36
5 Conclusions.....	37
5.1 Discussions.....	37
5.2 Future Work.....	40
6 References.....	41
7 Appendix A Abbreviations.....	45

LIST OF FIGURES

Figure 1: block diagram of the architecture of false alert reduction model using ISC algorithm.....	15
Figure 2 : a snapshot of the output of the query from Mysql which was saved in a file.....	19
Figure 3 : snapshot of the pre-processed and modified data file.....	21
Figure 4 : snapshot of the final output from the ISC filtering algorithm.....	24
Figure 5 : comparison of precision and recall for different number of training weeks (= strength).....	31
Figure 6 : relationship between false alarm filtering factor and threshold.....	33
Figure 7 : ROC curve for IDS alert filtering with ISC algorithm(threshold varied with 7.5, 7.0, 6.75, and 6.5 sequentially).....	34

LIST OF TABLES

Table 1 : time difference between U.S.A and Sweden at the time of data generation.....	22
Table 2: testing results for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(method: testing while stop training after the training phase).....	28
Table 3 : testing results for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(method: testing with all time training).....	28
Table 4 : no. of generated classes for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(for both types of testing procedures).....	29
Table 5 : testing results for three weeks testing data with same threshold value(6.50) for different strengthened training data(method: testing while stop training after the training phase).....	29
Table 6 : result comparison between two testing methods(with same threshold value(6.50) for different strengthened training data).....	30
Table 7 : Testing results for three weeks testing data with varying threshold value, having 10 times training data(testing while stop training after the training phase).....	32
Table 8 : comparison between the testing results where the target is a particular host and everything in the network jumbled together as a single target.....	35
Table 9 : result comparison between ISC filter and KNN filter.....	36

This master's thesis was performed by Champa Dey at **KTH** (The Royal Institute of Technology) within the **Department of Computer and Systems Sciences**. It was done as the integral part of a **master's degree in Information technology** within the specialization in **Information and Communication Systems Security**. The thesis was carried out at **SPOT** Lab(Security, Policy and Trust Laboratory) at **SICS**(Swedish Institute of Computer Science), Sweden.

The ISC algorithm has been applied in different application areas which are conducted as different projects by SICS. The same ISC algorithm was tested in this master thesis work as another application of the algorithm.

Champa Dey,
December 8, 2008
Stockholm, Sweden.

Abstract

Along with Cryptographic protocols and digital signatures, Intrusion Detection Systems(IDS) are considered to be the last line of defense to secure a network. But the main problem with todays most popular commercial IDSs(Intrusion Detection System) is the generation of huge amount of false positive alerts along with the true positive alerts, which is a cumbersome task for the operator to investigate in order to initiate proper responses. So, there is a great demand to explore this area of research and to find out a feasible solution.

In this thesis, we have chosen this problem as our main area of research. We have tested the effectiveness of using the Incremental Stream Clustering Algorithm in order to reduce the number of false alerts from an IDS output. This algorithm was tested with output of one of the most popular network based open source IDS, named Snort, which was configured to playback mood to look for DARPA 1999 network traffic dataset. Our approach was evaluated and compared with K-Nearest Neighbor Algorithm. The result shows that the Incremental Stream Clustering Algorithm reduces (more than 99%) the number of false alarms more than that of K-Nearest Neighbor Algorithm (93%).

Keywords:

Intrusion detection system, False positive alert, Incremental Stream Clustering algorithm, DARPA 1999 network traffic dataset, K-Nearest Neighbor algorithm.

Acknowledgment

At the onset, I am thankful to the Almighty God who blessed me to achieve this feat and I always owe to Him.

I would like to convey my heartfelt gratitude and appreciation to both of my supervisors at SICS(Swedish Institute of Computer Science) Tomas Olsson and Dr. Anders Holst, who guided me all the way to my thesis work. Without their constant support and guidance, it would not possible for me to successfully carry out this research.

I also like to thank my academic supervisor at KTH, Professor Dr. Louise Yngström for her kind assistance and for providing me enough freedom to carry out my thesis work. I am thankful to my thesis examiner, Dr. Oliver Popov for his kind suggestions to improve the thesis writing.

I am always indebted to my parents, Mr. Banita Mohan Dey and Mrs. Bithika Dey, for providing me the best studying environment always with constant love, support and encouragement. I was always inspired by my father's research activities from my very childhood who is an Mathematician.

I also like to thank my three younger sisters who have always been the source of encouragement and enthusiasm for my studies.

And, special heartfelt gratitude to my dearest husband, Mr. Subrata Sovan Deb, without whose immense love, support and sacrifice, it was not possible for me to carry out my higher study in Sweden.

Thank You!

1 Introduction

1.1 Background

An intrusion is a sequence of related actions performed by a suspicious adversary, which result in the form of compromise of a target system. These kinds of actions actually violate a certain security policy of the system. Security policy of a system defines which actions are considered to be malicious for the system and should be prevented in order to maintain the security of the system [15].

The process of identifying and responding to suspicious activities of a target system is called Intrusion Detection. It is an complementary approach to security with respect to the mainstream approaches, such as access control and cryptography [15]. Intrusion detection systems are used to monitor computer systems, as well as the network and to raise alarms when some intrusive activities are detected.

But most of the popular IDSs suffer from generating false alarms in a large volume. False alarms could be of two types. One is called false positive which is generated mistakenly by the IDS as an evidence of malicious behavior of the system, but in reality, it is not such a behavior. The other type of false alarms is called false negative. It is generated by the IDS as an evidence of non malicious event, but in reality, it should be an indication of malicious activity in the system [13, 14]. Previous research on this area reports that this value could be as high as several hundred thousands a day but around 99% of them are false alarms while monitoring intrusion in an active operational network [2, 6,19]. Network security officers need to investigate each IDS alarm manually whether it is a false or a true alarm. So, it is a quite time consuming, error prone and hard task for the network security officer to investigate manually and take proper action accordingly. Thus we have chosen to address the false alarm problem of IDSs in our research.

1.2 Problem

The effectiveness of an IDS depends on the capability to detect any abnormal activity in the target system, which is called the sensitivity of IDS. If the IDS is more sensitive, the security of the system would be tighter. Making the IDS more sensitive means to apply tighter signature rules or to be less tolerant to anomalies (information about 'signature rules' is provided in section 2.1). As a result, the IDS becomes more sensitive to its input and generates a lot of alarms each day, even though most of the examined events are not illegal events.

Due to large volumes of IDS false alarms, it is a quite tough task for the security officers to investigate manually which are the real suspicious alarms and thereafter take proper action against them. Even sometimes, some real suspicious alarms are ignored mistakenly by the security officer due to large volumes of false alarms and thereby mistakenly interpret a real alarm to be a false alarm. This is the most dangerous situation when a real instance of an attack is ignored by the security officer and thus the IDS becomes useless though its functionality remains the same.

We have chosen to investigate about this problem in our research and thus our research problem is whether we can reduce the IDS false alarm problem to a reasonable amount, or not.

1.3 State of Art

Due to the generation of large volumes of security audit data by the IDSs and the complex and dynamic properties of intrusion behaviors, the optimization of the performance of Intrusion Detection Systems (IDSs) becomes a very important open problem. Therefore, dealing with IDS alarms is one of the major research areas in intrusion detection.

To solve this problem, a wide variety of machine learning and data mining (details are provided in sections 2.2 and 2.3) techniques have been applied to intrusion detection which offer a good opportunity to improve the quality of output of an IDS and to facilitate easy administration of IDS.

In 2003, Julisch [6,16,17] used an alarm clustering technique which was used to group similar IDS alarms to a single cluster and to analyze IDS alarms to search for their root causes. He found that a few root causes of alarms are responsible for causing 90% of the IDS alarms. His clustering approach is one of the most successful alarm reduction methods for that time being and he was able to reduce 82% of false alarms without any false negative. However, this approach required manual processing to find out and also remove the corresponding root causes to produce those common false alarms

Pietraszek 2004[2] used an adaptive learner alert classification technique which uses background knowledge (for example, network topology) for IDS false alert reduction.

In 2006, Alshammari, Sonamthiang, Teimouri and Riordan [18] used a Neuro-Fuzzy approach to reduce the false positive alerts and they were able to reduce 90.92% of IDS false alerts.

Kwok Ho Law and Lam For Kwok 2006[1] used K-Nearest-Neighbor (KNN) classifier to filter IDS false alerts. We took their work as our reference work. Here is a brief description about how the KNN classifier works. The KNN classifier uses Euclidean distance to find out the similarity between two data points. The final similarity score of a data point being classified is the average of its Euclidean distance from the closest k normal points. If the similarity score is higher than a predefined threshold value, the point is classified as an abnormal point, otherwise it is assumed as a normal point. The same principle is applied in IDS false alarm filtering by the KNN classifier algorithm. First, they [1] modeled the alarm patterns of IDSs under an attack-free situation and, then detected anomalies from incoming alarm streams using KNN) classifier. They were able to reduce up to 93% of IDS false alerts by using the DARPA 1999 dataset [5], which is widely used as a public benchmark to evaluate and test the performance of an Intrusion Detection System. They used the first and third week of DARPA 1999 data set for the training purpose with the false alarm model and the second week's data as the testing data. Our approach resembles somewhat their approach in the modeling stage but differs in the underlying clustering algorithm, as well as, in procedures for the testing phase. We even took the data of second, fourth and fifth week of DARPA 1999 dataset as our testing data, whereas, they did the testing only on second week's of DARPA 1999 dataset.

Besides, alert correlation is another approach to handle IDS false alert, that is a process of collecting and relating alert information [5].

1.4 Hypothesis

Machine learning and data mining approaches have been applied in the area of intrusion detection in many studies. But, it is a relatively new area to apply machine learning and data mining for filtering false alarms from IDS output. We have reviewed the previous research works(mentioned above) on this area and believe that there exists more insights that data mining and machine learning techniques can contribute to this area of research. So, we concluded this research work in order to find a new method which can reduce more false alerts from an IDS output automatically, and thus reduce the work load of an security officer to investigate the IDS output. We have chosen a particular machine learning algorithm (the Incremental Stream Clustering algorithm) to be applied for IDS false alarm filtering purpose in our research, which has never been used earlier for solving such kind of problem. So, our hypothesis is that, the ISC algorithm could be used to filter out IDS false alerts automatically to some considerably large extent.

1.5 Goal

The goal of the thesis is to explore a methodology to reduce the number of false positives from an IDS output to a reasonable amount.

1.6 Purpose

The main purpose of this thesis is to reduce the work load of scanning the IDS output for the security officer by reducing the number of IDS false alarms automatically and thus enhancing the performance of an IDS.

1.7 Research Methodology

The type of our thesis is to develop a new artifact. The approach to solve our particular research problem was chosen to set up an experiment to test our explicit hypothesis, and thus empirically prove whether the hypothesis was true or not. So, the method of our thesis was deductive in nature at the logical level. Quantitative data were collected from the experiment and then, statistical data analysis was done to get the result.

We followed the below described methodology in order to achieve our particular research goal.

1. Literature Study – We studied and reviewed past publications and references related to this topic.
2. After that we analyzed the attempted solutions to the IDS false alarm problem and proposed our model of solution
3. Then we identified the technical and functional requirements for the proposed solution and the test case scenarios
4. According to the requirements, we installed specific softwares and configured them according to our requirements. Then we performed the pre-processing and re-programming of the code (the ISC algorithm), which was written for other applications.
5. Then we performed testing of the algorithm with a proper dataset (in our case, we used DARPA 1999 dataset) and with different test case parameters.
6. Finally we evaluated and compared our result with the referenced state of art.

1.8 Limitations

We did not test the ISC algorithm with real network data to filter IDS false alarms, because, we could not be able to find any public real data source which was suitable for this research.

1.9 Intended Audience

The thesis is expected to be beneficial for the network administrators or the security officers who are intended to investigate the output of IDS alarms. It is also beneficial for the IDS vendors, specially for the signature based IDSs, who can use it as an plug-in feature with their products.

1.10 Thesis outline

The remaining parts of the report are organized as follows:

Chapter 2 : provides a brief understanding about IDSs and some machine learning techniques that are used in the area of IDS alert filtering

Chapter 3 : provides an understanding about how the ISC algorithm works to filter IDS false alarms

Chapter 4 : gives the details about the experimental set up and the result of our test

Chapter 5 : concludes the findings we achieved from our experiment and suggests future work.

Acronyms are found in Appendix A.

2 Extended Background

2.1 Brief Details about IDS

In recent decades, intrusion detection has become one of the most complementary approach with respect to the main stream techniques in computer security field. The goal of this technology is to identify malicious activities from a stream of monitored data. This monitored data could be network traffic, operating system events or log entries. An IDS can be composed of several components: **Sensor/s**, which detects security events, **Console**, which is used to control the sensor/s and to monitor the security events and alerts, and a central **Engine** which records the events generated by the sensor/s and produces alerts according to certain predefined heuristics [20].

Now-a-days, there are different types of IDSs depending on the type and location of the IDS sensor and on the methodology/heuristic used by the engine to generate alerts.[20] All different detection methodologies can be categorized as signature based detection (which is also called misuse detection) and anomaly based detection (a.k.a. behavior based detection) [15].

Anomaly-based detection is based on the assumption that an intrusive or anomalous behavior has some apparent different characteristics from a normal one. It assumes that normal activities are in majority and suspicious activities in minority in a system[1,4]. So, it relies on the base that detecting certain deviations of behavior from the majority would be able to detect any malicious activity. So, first, an anomaly based system builds a model of normal behavior of the system, and then, it searches for anomalous activities in the system which are not recognized by the model. If the anomaly based system finds evidence of any of such suspicious activities, it raises a flag, that is an indication of intrusion to the system. The modeling can be accomplished in several ways. These include statistical modeling and artificial intelligence type techniques. However, a problem with anomaly detection approach is that, it produces comparatively large deal of false positives, though, it is able to detect new type of attack/ anomalous behavior, which is sometimes called “zero days attack” [15].

A majority of current intrusion detection systems (IDS) use a signature-based approach in which, events are detected that match specific pre-defined malicious patterns or attack models, known as "signatures". The signatures are saved on a database. The data collected by the IDS is compared with the content of the database. If a match is found, an alert is generated as an evidence of system compromise. The events that do not match any pre-defined malicious pattern, are considered to be the legitimate activities. It produces fewer false positives than the anomaly based detection. In order to function efficiently, the signature-based IDS needs to update its attack model database timely [15,20]. But the main limitation of signature-based IDS is their failure to identify new attacks, and sometimes even minor variations of known patterns if their knowledge database is not properly updated. Besides, a significant administrative overhead is needed to maintain signature databases. Example of a popular signature-based IDS is Snort.

To take advantage of both of these approaches, some researchers have worked on a hybrid approach.[15]

Depending on the placement of the IDS, it is categorized into three categories [15]:

- 1) Network-based IDS (NIDS)
- 2) Host-based IDS (HIDS)
- 3) Application-based IDS

A network-based IDS focuses on the network packets over a network (where the IDS is installed) and analyzes them to detect whether any intrusion has occurred in the specified network[15,20]. Example of a network-based IDS is Snort.

A host-based IDS looks at the audit data of a specific host(where the IDS is installed) and watches whether the activities on the host computer are intrusive or not. These audit data are produced by the host operating system. Audit sources include system information and syslog facility [15] Example of a host-based IDS is OSSEC [21].

Application-based IDSs detect attacks in a specific application of a system [15].

2.2 Machine Learning

Machine learning is the sub-domain of artificial intelligence which is concerned with developing methods for a machine (e.g., a computer) to learn from experience or extract knowledge from examples in a database [23, 24]. It offers a major opportunity to improve quality and to facilitate administration of IDS. There are a variety of machine learning algorithms, but, most of them fall under the following classes [25]:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

2.2.1 Supervised Learning Algorithm

Supervised learning algorithm produces a function that maps inputs to desired outputs. Actually it learns the function from the training examples which consist of pairs of input objects, and desired outputs. One standard formulation of the supervised learning task is the classification problem. In a classification problem, the learner needs to learn and approximate the behavior of a function, which maps an input vector into one of several classes, by looking at several input-output examples (i.e., labeled data) of the function [24, 26].

2.2.2 Unsupervised Learning Algorithm

Unsupervised machine learning algorithm models a set of input where labeled examples are not given [24, 27]. In a sense, unsupervised learning can be thought of as finding patterns in the unlabeled input data. A very simple example of unsupervised machine learning is clustering problems. However, the application of unsupervised learning technique in the intrusion detection area can detect new kinds of attacks, provided, the attacks exhibit some unusual character in some feature space.

2.2.3 Reinforcement Learning

Reinforcement learning is a area of machine learning, which is concerned with learning a policy of how to interact with its environment, given an observation of the world. It relies on the fact that, its every action has some impact in the environment, and the environment provides feedback which guides the learning algorithm [24]. Reinforcement learning is closely related to the fields of control theory (in engineering), and to the decision theory (in statistics and management science) [32].

2.3 Data Mining

Data mining refers to the process of extracting useful patterns from a huge volume of data by using specialized algorithms [1, 22]. It plays an important role in the area of information retrieval. Specialized algorithms are mainly based on the machine learning techniques. In other sense, data mining is an application of machine learning techniques.

2.4 Machine Learning in Intrusion Detection

Several different types of machine learning techniques have been using in the area of intrusion detection since 1990. Here, we discussed about some machine learning techniques which have been used in this area of research.

2.4.1 Classification

Classification is a supervised machine learning technique which assigns one of several pre-defined class labels to data objects. The classifier needs to be trained with labeled input examples, so that it could understand the characteristics of different classes, and then, it could be able to map new data items to different classes [1].

2.4.2 Clustering

Clustering is the process of distinguishing and sorting similar types of objects into similar groups, in such a way that objects from the same cluster are more similar to each other than objects from different clusters. The similarity between the clusters is measured according to a parameter, called 'distance' measurement. It is an important step in any clustering is to select a distance measure, which will determine how the *similarity* of two elements is calculated. It influences on what will be the shape of the clusters, as some elements may be close to one another according to one distance measure and, farther away according to another [28].

2.4.3 Artificial Neural Network

An artificial neural network is a mathematical model that is based on biological neural network. Artificial neural networks are non linear statistical data modeling tools which can be used to model complex relationships between inputs and outputs, in order to find patterns in the data. It consists of an interconnected group of artificial neurons(which are actually mathematical functions) which are used to process information. An important characteristic of artificial neural network is that, during the learning phase, it can change its structure based on external or internal information that flows through the network [29].

2.4.4 Bayesian Classification

Bayesian classification is an unsupervised machine learning approach which uses Bayesian inferences. It does not divide the given data into classes, but rather, it defines a probabilistic membership function of each datum into the most likely determined classes [29]. The result of Bayesian learning is expressed in terms of probability distributions over all unknown quantities.

3 Our Proposed IDS False Alarm Filter with ISC Algorithm

3.1 Incremental Stream Clustering (ISC) Algorithm

Anomaly detection refers to separating patterns in a given data set, which deviate in a certain prominent magnitude from the regular majority of patterns. The Incremental Stream Clustering algorithm is a Bayesian Anomaly detection algorithm, which uses unsupervised machine learning technique. More specifically, the underlying methods are statistical methods [3], which basically are Bayesian inferences for finding expected distributions for clusters of patterns and then using these distributions for classifying and assessing a degree of anomaly to a new pattern. So, the implementation of ISC algorithm to reduce IDS false alerts has two phases. They are:

1. Training phase
2. Testing phase or anomaly detection phase

3.1.1 Training phase of ISC algorithm

In the training phase, data is collected from a large number of (predominantly) normal situations for a certain time period, and a statistical cluster model is estimated from these data. This model defines what is considered to be normal. Then it uses the model to classify a new pattern correctly as either belonging to a certain previously observed cluster or not belonging to any of these already generated clusters. If the pattern is not likely to fit any of the already created clusters then a new cluster is created for this single pattern. In a sense, we can say statistically that some sort of aggregation of similar properties is done during the clustering.

In our case, an observation is a set of data points in a multi-dimensional space which represents the frequencies of different alarm types during a limited time period. When a new observation is obtained, the probability that the estimated model should generate this observation is calculated. If the probability is sufficiently low, i.e., the observation is highly anomalous with respect to all the generated clusters, the observation is considered to fall into a new cluster. In this way, a new cluster is being generated. On the other hand, if the estimated probability is not sufficiently low, the observation is classified into the most probable cluster, where it is not anomalous. The cluster model is updated to reflect the new member. Since this probability is often very close to zero, it is common to deal with the negative log-likelihood, i.e. minus the logarithm of the probability of generating the observation, which is then a large positive number [4].

To assess how anomalous an observation is, we used the formula below. Given an observation X (i.e., a pattern) with the probability density distribution $p(x)$ of a cluster, the deviation $Dev(X,z)$ of a pattern z from X is defined as,

$$Dev(X,z) = \frac{E[\log p(X)] - \log p(z)}{S[\log p(X)]}$$

where,

$$S[\log p(X)] = \sqrt{Var[\log p(X)]}$$

The deviation is the negative log likelihood normalized to have an expected value of zero and standard deviation of one. As a rule of thumb, this means that the deviations between -3 and +3 are “just normal”, and even values up to 6 may occur occasionally just by chance. This also means that the threshold for considering an observation to be an anomaly should usually be well above 6.

3.1.2 Testing phase or anomaly detection phase of ISC Algorithm

In the testing phase, data is collected from a large number of (predominantly) abnormal situations for a certain time period, where it contains attack instances. The data is tested to find out the existence of abnormal patterns of data, which is calculated from the estimated normal cluster model from the training phase. If a new observation is anomalous, an alert is raised for further investigation by the administrator. In this experiment, it is considered to be an indication of an attack. Otherwise it is considered to be a normal point.

There are two alternative approaches in the testing phase. They are described below:

3.1.2.1 The First Approach

In this testing approach, the training is stopped after the training phase and thereby, no new clusters are generated in the testing phase. During this phase, when a new observation comes, the following considerations are taken in to account for anomaly detection. For a given time window, the pattern of the observation is anomalous if the minimum deviation, with respect to all existing clusters, is greater than a predefined threshold.

3.1.2.2 The Second Approach

In this testing approach, the testing is done simultaneously without stopping training after the training phase. In other words, in this approach, continuous training is done along with the testing procedure. Unlike the previous approach, new clusters are generated during this testing phase. These new clusters are seemed to be anomalous clusters, since the testing dataset contains some attack instances. Here, when a new observation comes, the following considerations are taken in to account for anomaly detection. For a given time window, the pattern of the observation is anomalous either where, the minimum deviation, with respect to all existing clusters, is greater than a predefined threshold or, the observation is a member of a new cluster which is generated during the testing phase.

3.2 Running Time of the ISC Algorithm

Running time is an important parameter for measuring the performance of any algorithm. In our case, the running time of the ISC algorithm is a linear function with respect to the number of input attributes in an observation, the number of generated clusters(which is usually small), and the number of observations to classify. In practice, the running time of the algorithm was not a problem at all.

3.3 Incremental Stream Clustering Model for False Alert Reduction

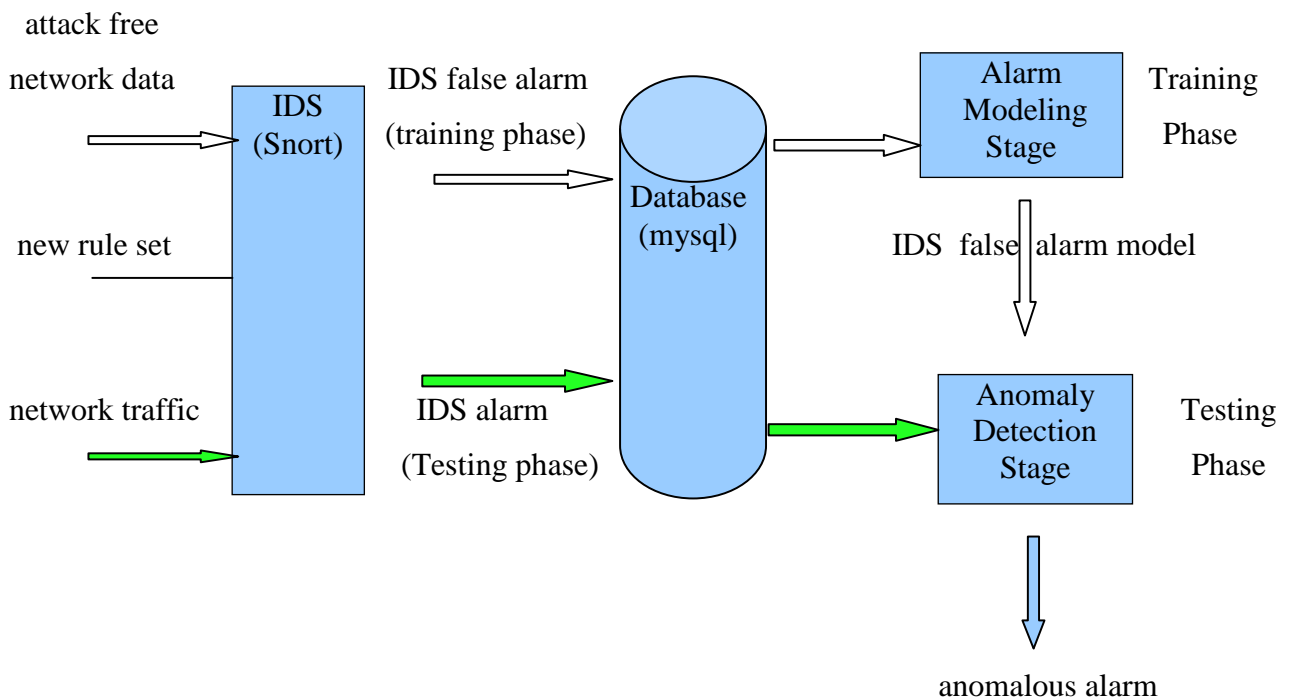


Figure 1: Block diagram of the architecture of false alert reduction model using ISC algorithm

The above figure (figure-1) shows the conceptual architectural block diagram of our alert filtering process. The algorithm has been implemented in two stages:

1. Alarm modeling stage where the clusters are built in attack free situation in the training phase
2. Alarm filtering stage or Anomaly detection stage for testing phase

During the training phase, it is assumed that the input dataset is free from any attack, or any suspicious event. So the clusters, which are generated in this training phase, are considered to represent different normal situations. These clusters are treated as the normal pattern of behaviors of the alarms, where there is no suspicious activity.

3.4 Implementation of the ISC Algorithm

The algorithm was previously implemented for other applications in C++. We chose to use Java for our application as the programming language and that is why an Java wrapper was written which makes the C++ API available in Java using JNI technology. Finally for the filtering purpose, we added another Java class along with the Java converted algorithm.

4 Testing and verification of the ISC Algorithm

4.1 Data Source

We used Snort [8](version: 2.8.0.2) as our preferred IDS which is a popular and widely used open source signature based network IDS. To conduct the testing, we downloaded the DARPA 1999 off-line intrusion detection dataset [5] from MIT Lincoln Lab's website. This dataset contains simulated network traffic embedded with marked attacks. The data was collected from a simulated medium-sized computer network in a fictitious military base. The network was connected to the outside world through a router. The policy of the router was set such that, it would not block any connections.

The simulation took place during five weeks (five days per week), which yielded the first three weeks for training data and the last two weeks for testing data. The first and the third week's data contains no attacks. The second week's data is kept there for training with labeled known attacks. The fourth and fifth week's data are for testing dataset, which contains new and some old attacks. The DARPA 1999 dataset consists of: two sets of network traffic data (inside Tcpdump data and outside Tcpdump data), audit data (BSM and NT), and directory listing. MIT Lincoln Lab has also provided the attack truth tables which is a description of the attacks that took place in the simulated data. This table is also called the ground truth table.

4.2 Experimental Setup

For performing the experiment, we ran Snort on a Linux machine (Linux 2.6.22.14-72.fc6) equipped with Intel Pentium 4.0 CPU 3.00 GHz processor and 1.0 GB RAM. We used the default configuration and the latest rule set (for that particular version) because our intension is not to evaluate or enhance Snort's performance as an IDS, but rather to reduce the number of false alarms generated by Snort. We configured Snort in batch mood to replay the DARPA 1999 Tcpdump data. The following command was given to run Snort with -r switch to reply the Tcpdump data:

```
snort -r inside.tcpdump -c /etc/snort/snort.conf
```

In our experiment, we chose to investigate the anomalous behavior of the hosts of MIT Lincoln lab's internal network. That is why we chose to play only with those data, which were triggered by the sniffers placed inside the lab. So, we replayed the inside Tcpdump data to Snort. We also configured Snort in such a way that the output of the Snort alarms was saved in a database. We used Mysql for this purpose. We created 19 tables in the Mysql database that contained several information regarding the Snort alarms. Then we used a query from two of those tables (EVENT table and IPHDR table) in order to extract the value of timestamps, event ID, signature name, source IP and destination IP for each alarm. These pieces of data would be used for statistical modeling and classification. We ran the following query command on Mysql to extract these values and saved the result in a file:

```
select iphdr.cid as event_id, event.signature as signature_id, TIME_to_sec(event.timestamp)  
as timestamp_in_second, event.timestamp as timestamp, inet_ntoa(iphdr.ip_src) as source_ip,  
inet_ntoa(iphdr.ip_dst) as destination_ip from iphdr inner join event on iphdr.cid=event.cid;
```

A snapshot of the file which stored the output of this query is as follows:

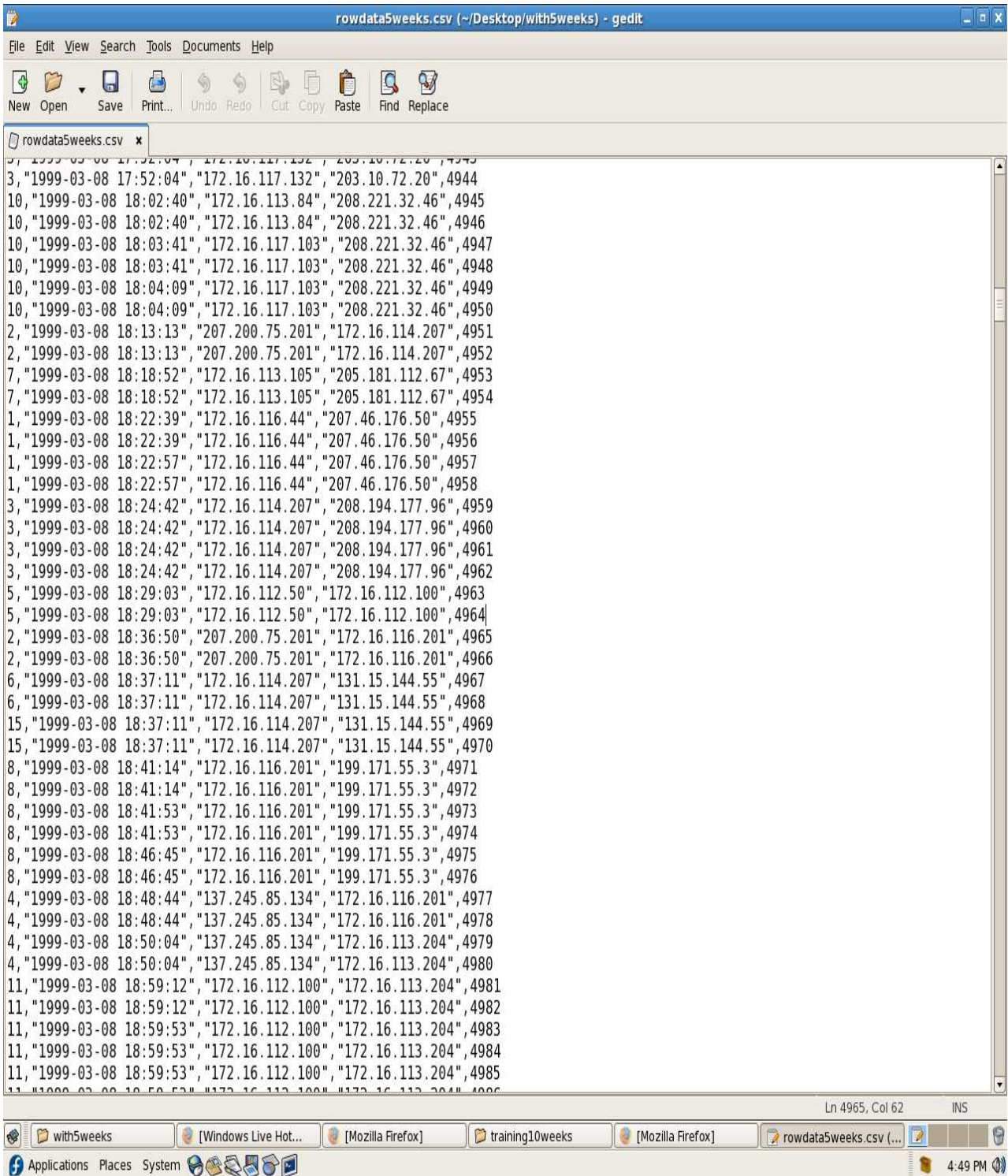


Figure 2 : A snapshot of the output of the query from Mysql which was saved in a file

Since the internal hosts of the MIT Lincoln's Lab were our main concern, we took the corresponding private IP of those machines as our target(where we would look for an investigation) for each alarm, irrespective of whether it was a source or destination machine. For creating the model we used three different parameters of an alarm: timestamp, signature name and the corresponding IP of inside host. So, here arise some considerations for a single alarm in a particular period:

a) When one of the source IP or destination IP is from the outside network, the IP of the outside host is discarded and corresponding timestamp, the signature name and the IP of the inside host is considered for that alarm

b) When both the source IP and destination IP are from the inside network, it creates problem because, we have to consider both hosts, irrespective of source or destination of an attack. In this situation, we made two instances of alarms from the same single alarm, because, both inside hosts should be investigated equally. So we made two instances of alarm, keeping the same timestamps for both of the inside hosts and the signature name. In this addition, we also wanted to make the separation between the source signature and destination signature, since they contain different data. But we only had one signature name for both of the inside hosts. So, to deal with this consideration, we added extra information to the signature name for each alarm. If the host was a source, the signature name was concatenated with “_source” and if the host was a destination, the signature name was concatenated with “_destn”. This was also done while there was only one host from source and destination in order to cope with this new issue.

The following is an example from our data file.

<u>Timestamps</u>	<u>IP from inside host</u>	<u>Signature</u>
9012333422	172.16.14.20	11_destn
9012333423	172.16.18.51	32_destn
9012333427	172.16.14.20	5_source
9012333428	172.16.14.50	11_source
9012333428	172.16.14.100	11_destn

c) Another consideration was regarding day light saving time issues where and when the data was generated and where it was manipulated and tested. The data was generated in USA from 15th March,1999 to 10th April, 1999. The manipulation on that data was done in Sweden for this research work. Because of the differences in the day light saving time in Europe and in USA, we needed to adjust the time difference between these two countries for that time. We took into account the exact time difference between these two places for that period for the perfect evaluation of our result. The time difference between these places for that period is given in the following table-1:

Time	Time difference
15 th March,1999 to 27 th March,1999	6 hours
28 th March,1999 to 2 nd April,1999	7 hours
5 th April,1999 to 10 th April,1999	6 hours

Table 1 : time difference between U.S.A and Sweden at the time of data generation

4.3 Testing with DARPA 1999 Dataset

We trained our alarm model with the first and third week of DARPA 1999 dataset which contain no attacks. Though the second week of DARPA 1999 dataset contains some labeled known attacks, MIT Lincoln Lab has considered them as data for the training purposes. For a supervised learning technique it is important with training examples of attacks, but this is not necessary for an unsupervised method. The reason behind this is that, for the learning purpose, the unsupervised machine learning technique does not need to match the input with a predefined or desired output. In this technique, it learns from the unlabeled inputs.

Kwok Ho Law and Lam For Kwok [1] used this second week's data as their testing dataset. (They did not use week 4 and week 5 as their testing dataset).But, as these are simulation data and not the real data, we found that, in second week's data, there were more attack instances than the normal data(From Snort's output for this week, there were total of 15,362 attack instances among 17614 of total alerts). So, testing with only second week's data is not perfectly suitable for the testing purpose and evaluating an algorithm's performance. That is why, we took the data of second week, fourth week and fifth week as our testing data.

A data point in the ISC model represents the alarm distribution of different signature types in a certain time period. We took 1 hour as the the time window and the sampling rate (i.e. Stepping for outputting) was 30 minutes. It means that we constructed a new data point in every 30 minutes, containing the signature counts for the last 60 minutes, as an input to the ISC algorithm.

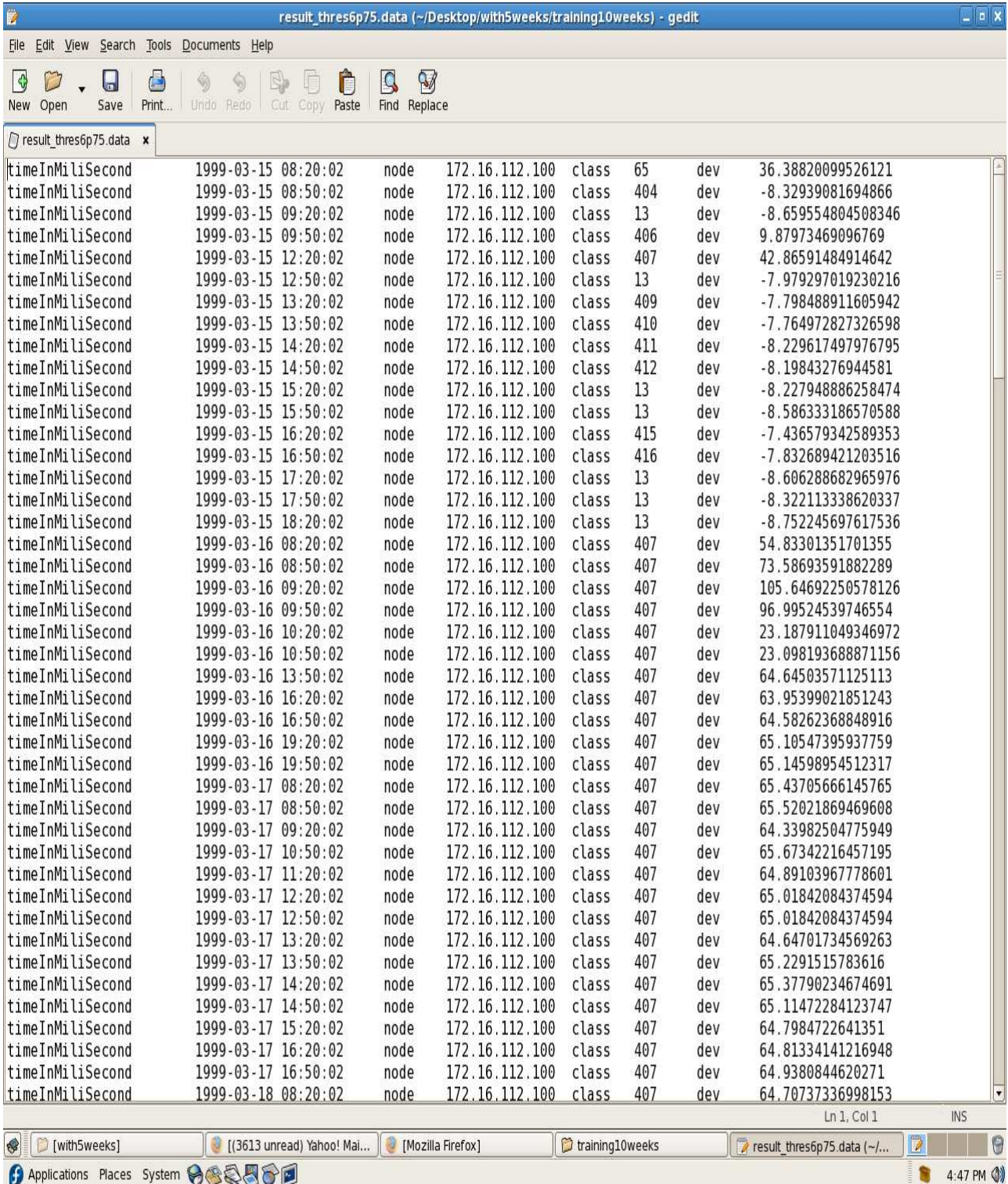


Figure 4 : snapshot of the final output from the ISC filtering algorithm

4.4 Measurements for Experimental Validation

We used the below mentioned measures for the experimental validation purpose for our research. Here, we have referred to alerts related to attacks as 'true positives', alerts triggered mistakenly by benign(i.e., non malicious) events as 'false positives', alerts related to non attacks as 'true negatives', and alerts related to attacks but triggered mistakenly as benign(i.e., non malicious) events, as 'false negatives'.

$$\text{True Positive Rate} = \frac{\text{no. of true positives}}{\text{total number of positive instances}}$$

$$= \frac{\text{no. of true positives}}{\text{no. of (true positives+ false negatives)}}$$

$$\text{False Positive Rate} = \frac{\text{no. of false positives}}{\text{total no. of negative instances}}$$

$$= \frac{\text{no. of false positives}}{\text{no. of (false positives+ true negatives)}}$$

In an Information Retrieval scenario, Precision is defined as the number of relevant documents retrieved by a search, divided by the total number of retrieved documents. “Recall is defined as the number of relevant documents retrieved by a search, divided by the total number of existing relevant documents (which should have been retrieved)”[31]. For our experiment, the Recall and precision is defined as:

$$\text{Recall} = \frac{\text{no. of true positives}}{\text{no. of (true positives+ false negatives)}}$$

$$\text{Precision} = \frac{\text{no. of true positives}}{\text{no. of (true + false) positives}}$$

Two important measures of IDS false alarm filter are the reduction rate and the filtering factor. Reduction rate measures, to what extent IDS false alarms are filtered, and, it is calculated in percentage. On the other hand, filtering factor compares the output of IDS false alarm filter to its input data, which is provided by an IDS.

$$\text{Reduction Rate} = \frac{(\text{total no. of IDS alerts}) - (\text{total no. of alerts from the filter})}{\text{total no. of IDS alerts}}$$

$$\text{Filtering Factor} = \frac{\text{total no. of IDS alerts}}{\text{total no. of alerts from the filter}}$$

4.5 Results

The number of alarms collected by Snort from the DARPA 1999 inside Tcpcdump data for five weeks is 37772, among which, only 4276 are the alarms from the two weeks training data and the rest are from the three weeks testing data. Snort was able to detect 74 different attack instances in these three weeks of data in our particular settings.

First, we tested our ISC algorithm training with the original two weeks of training data tuning with different threshold values. We found the optimum threshold value to be 6.75 for this training dataset and then watched the testing result. We followed both of the procedures in the testing phase: testing with all time training and testing without all time training (training only in the training phase). Finally, we checked the output from our experiment with the ground truth (description is provided in section 4.1) provided by MIT Lincoln Lab for DARPA 1999 dataset. The result of detection of the attacks was not that much promising for both of the cases. Then we investigated what could be the possible reason for this unsatisfactory result. We guessed that the number of training examples were not sufficient enough to train the classifier perfectly. In order to justify our assumption, we manipulated the training dataset by artificially multiplying the data 5 times and then performed the same testing procedure with the same threshold value. It is to be noted here that, the artificially created data were kept sequential in time. However, we then found that, this result is much better than the former one and thus, it justified our previous assumption. The comparative result is given below (table-2 and table-3):

No. of Training repetitions (strength)	Total number of filtered output	Total number of true attack instances detected	Total number of true attacks detected	Recall (True Positive Rate)
1 time	101	79	44(among 74)	0.5945
5 times	160	108	60 (among 74)	0.8108

Table 2 : testing results for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(method: testing while stop training after the training phase)

No. of Training repetitions (strength)	Total number of filtered output	Total number of true attack instances detected	Total number of true attacks detected	Recall (True Positive Rate)
1 time	136	98	51(among 74)	0.6891
5 times	157	113	62(among 74)	0.8378

Table 3 : testing results for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(method: testing with all time training)

We also found that a better result in terms of the number of generated normal clusters and anomalous clusters, while we have much training examples, (i.e, when we artificially multiplied the training dataset and then performed the testing). Here is the result below (table-4):

No. of Training repetitions(strength)	No. of created classes (method: testing while stop training after the training phase)	No. of normal clusters (method: testing with all time training)	No. of anomalous clusters(method: testing with all time training)
1	11	11	40
5	34	34	49

Table 4 : No. of generated classes for three weeks testing data, with same threshold value(6.75) for two different strengthened training data(for both types of testing procedures)

So, we chose not to perform the testing with the original strengthened training dataset. In an attempt to improve the performance of the algorithm again, we repeated the training data 5, 10 and 15 times, thus we artificially multiplied the amount of training data into 10, 20 and 30 weeks, instead of two weeks of original training data and compared the results. Here, the optimum value of the threshold for the testing was found to be 6.50 for all these manipulated data. The result of testing with three different sized training data with the same threshold value (= 6.50) is given in table-5.

No. of Training repetitions (strength)	Total number of filtered output	Total number of true attack instances detected	Total number of true attacks detected	Recall (True Positive Rate)	Precision	Reduction Rate
5 times	160	108	60 (among 74)	0.8108	0.6750	99.52%.
10 times	192	121	67(among 74)	0.9054	0.6303	99.43%
15 times	216	137	70(among 74)	0.9459	0.6342	99.36%

Table 5 : testing results for three weeks testing data with same threshold value(6.50) for different strengthened training data(method: testing while stop training after the training phase)

We also investigated which of the two testing methods is better than the other. We found that, testing with all time training approach gives better result when we test with comparatively smaller amount of training data. But, when we gradually increase the training instances, the other method (i.e, testing while stop training after the training phase) gives better result in terms of detecting the true attacks. The result is given in the following table-6:

No. of Training repetitions (strength)	Total number of true attacks detected (method: testing with all time training)	Total number of true attacks detected (method: testing while stop training after the training phase)
5 times	64 (among 74)	60 (among 74)
10 times	66(among 74)	67(among 74)
15 times	68(among 74)	70(among 74)

Table 6 : result comparison between two testing methods(with same threshold value(6.50) for different strengthened training data)

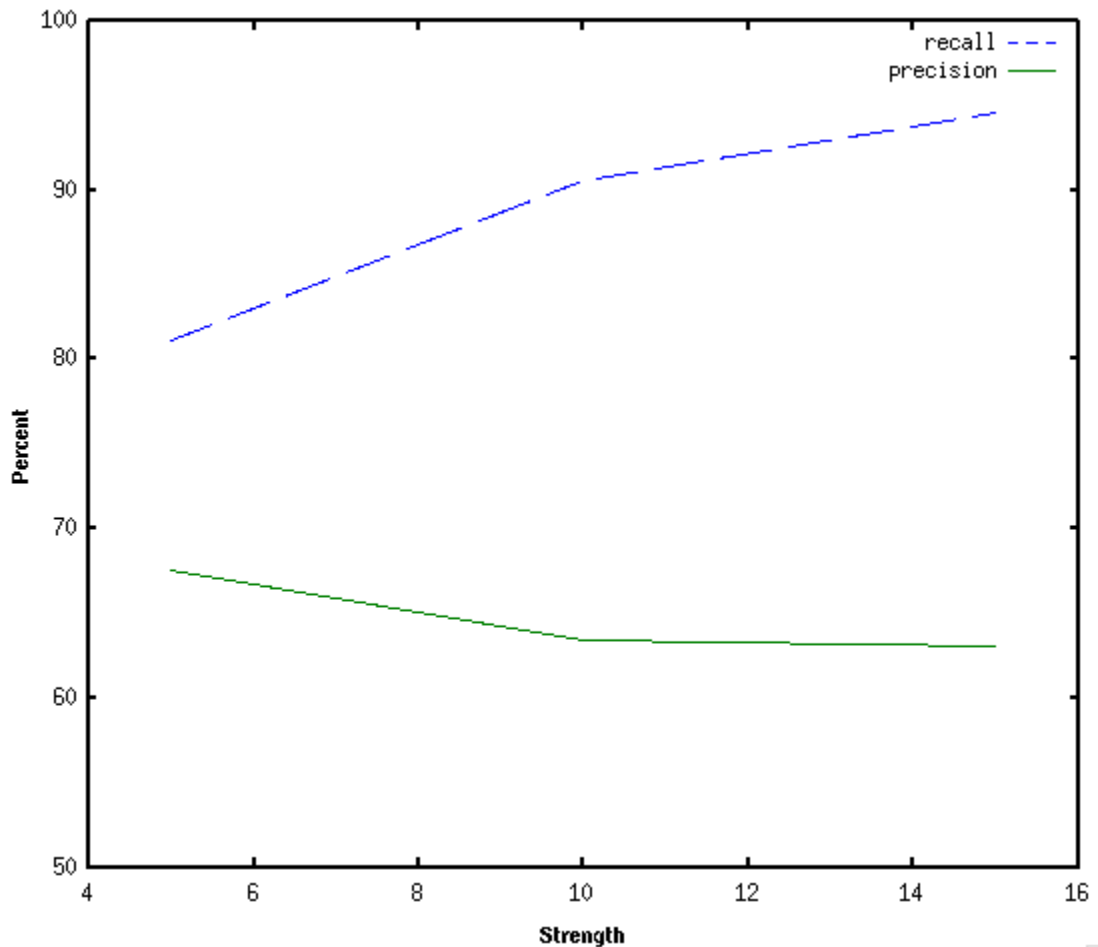


Figure 5 : Comparison of precision and recall for different number of training weeks (= strength)

In the above graph (figure-5), the performance of the ISC algorithm to filter out IDS false alarms is drawn according to training with different size of training data. We can observe that, the performance of the algorithm to filter out IDS false alarms gets better result with respect to large amount of training instances.

Tested threshold	Total number of filtered output	Total number of detected true attack instances	Total number of true attacks detected (among 74)	Recall (True Positive Rate)	Precision	False Positive Rate	Reduction Rate	Filtering Factor
6.5	192	121	67	0.9054	0.6303	0.3698	99.43%	175
7.0	171	117	65	0.8784	0.6842	0.3158	99.49%.	196
7.5	166	116	65	0.8784	0.6988	0.3012	99.50%	200

Table 7: Testing results for three weeks testing data with varying threshold value, having 10 times training data(testing while stop training after the training phase)

In the above table-7, the testing result of the algorithm for filtering the IDS false positives, are given for varying threshold values but with same amount of training data(10 times than the original training data). We can see that with higher value of the threshold, the reduction of false alarms increases but the true positive rate decreases. That means, the increase of the value of the threshold makes the algorithm miss more true alerts, though it can reduce more false alerts. The reason is, a high threshold value would make the algorithm more restrictive regarding what is considered an abnormal observation. Thus less observation will be judged as abnormal. So, the security could be looser and some noticeable alarms could be missed in this case. So, the value of the right threshold for testing our algorithm's performance must be chosen carefully. For this purpose, we tested our algorithm tuned with different threshold values and finally chose one threshold value for the evaluation purpose.

Alarm reduction is desirable only if a certain level of security is maintained. That is why, we did not choose such high value of threshold which would make the algorithm filter too many true alerts, though we missed some of the true alerts in fourth and fifth week's testing data. The following graph (figure-6) shows the relationship between the false alarm filtering factor and the given threshold values.

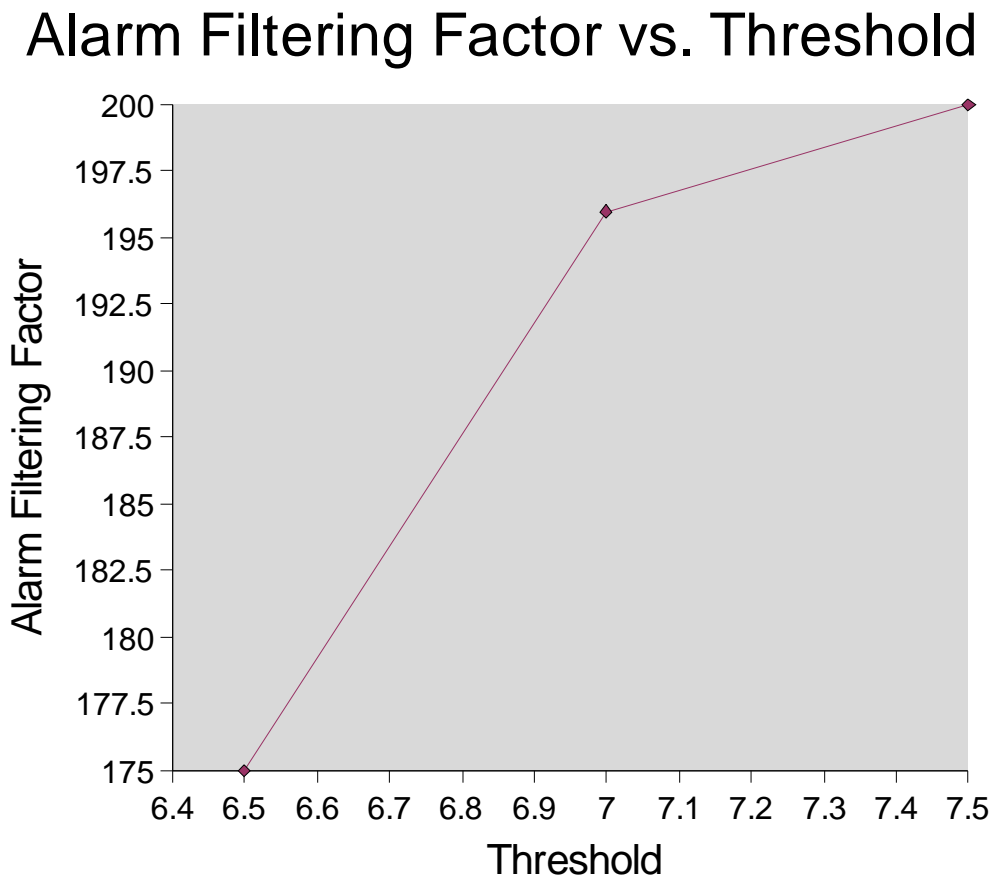


Figure 6 : Relationship between false alarm filtering factor and threshold

We draw the following ROC (Receiver Operating Characteristic) curve (figure-7) with our experimental results, which show the performance of the system under different settings with varying threshold value. ROC curve is a graphical representation of sensitivity versus (1-specificity) of a system when its discrimination threshold is varied [8]. ROC curve is represented by plotting the fraction of the True Positive Rate versus the False Positive Rate. From the graph (figure-) we can observe that, both the detection rate (i.e., true positive rate) and the false positive rate are high while testing with lower threshold value (for example, 6.5 as the threshold). Both, the true positive rate and the false positive rate decrease accordingly while testing with successive increasing values of the threshold. The following graph is the output of testing with the threshold values of 6.5, 6.75, 7.0 and 7.5 respectively.

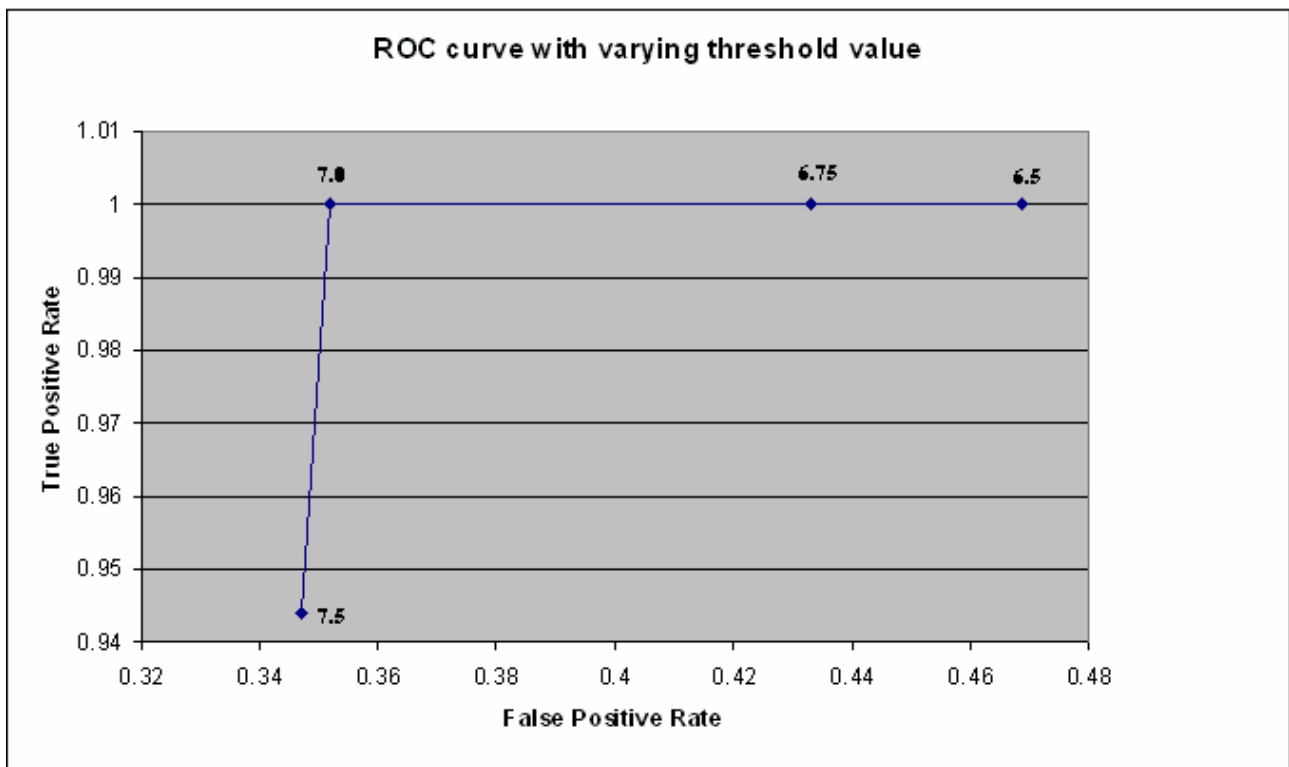


Figure 7 : ROC curve for IDS alert filtering with ISC algorithm(threshold varied with 7.5, 7.0, 6.75, and 6.5 sequentially)

It should be mentioned here that, in our research, the implementation of the ISC algorithm can be configured in two different ways to look for the anomalies and filter out IDS false positives. Either it can be configured to look for the anomalies in a particular host, or, look for the anomalies where everything in the corresponding network is jumbled together as a single target. We tested both of the alternatives and found that, testing with configuration where everything in the corresponding network is jumbled together as a single target, gives much better results than testing a single host as a target. For example, for testing with threshold 7.0 , 10 times training repetitions and method as testing while stop training after the training phase, we got the following result (table-8) for detecting attacks in a particular host(private IP : 172.16.114.50). We used the jumbling together configuration for all of our testing given in this report.

Total number of detected attacks for host 172.16.114.50, while partition is a particular host	Total number of detected attacks for host 172.16.114.50 while everything in the corresponding network is jumbled together as a single target
12	26

Table 8 : comparison between the testing results where the target is a particular host and everything in the network jumbled together as a single target

4.6 Strength of the ISC Algorithm against KNN Algorithm for filtering IDS alarms

We also tested our filter according to the testing procedure followed by Kwok Ho Law and Lam For Kwok[1], where they tested only with the second week's data of the DARPA 1999 dataset, and, compared our result with their result. We found that the our algorithm provides better result than that of KNN classifier (table-9).

Algorithm used	Detection Rate	Filtering Rate
KNN	100%	93%
ISC	100%	99.71%

Table 9 : Result comparison between ISC filter and KNN filter

5 Conclusions

5.1 Discussions

Our experiment shows that the ISC algorithm can achieve more than 99% reduction rate for IDS false alarms when tested on the DARPA 1999 data set whereas the referenced work [1] is able to reduce 93% of the IDS false alarms with the same dataset. In that sense, we can particularly conclude that the efficiency of the ISC algorithm to filter out IDS false alerts is highly appreciable. So the finding of this research can be used as an add-on to any IDS for alert filtering purpose. The filtered output from the implementation of ISC algorithm can be sent to the security officer for further investigation. Once the anomaly detection part of the implementation of ISC algorithm has issued an alarm, it is an easier task to go through and verify that it was an attack. After all, the aggregation of alarms during the last half hour is there (since we constructed a new data point in every 30 minutes), and the anomaly detection part may tell which signature had a suspicious amount of alarms (details are provided in section 3.1.1, 3.1.2, 4.2 and 4.3), so the security officer will actually know exactly which alarms to look at.

Though we had very few false alarms in our output, we suspect that some of these false alarms could be real true alarms which were not detected by Snort. Snort only looks for sign of attack from the incoming traffic through finding a match in its own attack signature database and if it finds a match, it asserts an alarm [8]. If the signature database is not updated and there is a new type of attack, Snort is unable to detect it. But, it is possible that the ISC algorithm could be able to find the traces of an attack which is undetected by Snort. The reason behind this fact is, an attack instance does not only take place between the attacker and the attacked host/hosts, but it also has some consequences on other hosts as well as on the entire network. So, the ISC algorithm can detect an anomaly looking on other signatures produced by other hosts at the same time an attack was

introduced which was undetected by Snort. For instance, we found an instance of an attack during the time interval between 18.20 and 18.50 on 1999/04/01, for all of our testing results and we can see from the attack ground truth table that there is a real attack on a host at 18:32:17 for a duration of 10:07 minutes on that day. But, this attack was not detected by Snort. So, we considered it to be a false positive. There were several such cases in our experiment when the ISC algorithm detected an attack which was not detected by Snort but which showed evidences of an attack at that time in the attack ground truth table. We did not investigate more on this matter during our thesis work but, we considered this fact as a future research domain.

In our experiment, we did not take into considerations about the changes in the network (e.g., adding new sub networks), user behaviors (e.g., changes in the size of a user group) or IDS configuration (for example, signature rule set updates). These are very important issues while modeling a network behavior perfectly, especially when we model a real network. If these are not considered while modeling a real network, some real attacks could be missed by the anomaly detector and thus the reduction of false alarms could gradually be declined. Since we used the DARPA 1999 dataset for the testing purpose, which actually contains simulated data, these issues are not considered as having a large impact on our result.

The ISC algorithm can be used in other ways to IDS false alarms filtering. For example, we could test the algorithm using a supervised learning approach. We also could use this algorithm for pure intrusion detection with raw network traffic. As the ISC algorithm is able to detect any abnormal behavior of a system from its known normal behavior, it can be used as the anomaly detection algorithm for an anomaly based IDS.

The reduction of the IDS false alarm can be viewed as a typical classification problem which is a typical research area in the domain of data mining. So, the same research goal can be also obtained by other popular classification algorithms like decision tree, artificial neural networks, etc. In that case, the modeling, the testing procedure, as well as the testing result could be different from ours but, the same research goal can be achieved.

The DARPA data set has been questioned in the research community for years because of its well-known weaknesses. On the basis of published descriptions of the data generation process of the DARPA dataset, McHugh's [9] primary criticism of the evaluation was regarding the failure of verification that the network realistically simulated a real-world network. He also criticized that the data rates were far below what will be experienced in a real medium sized network and the dataset does not model sufficiently advanced attacks that require more advanced detection systems. In 2003, Malhony and Chan [10] investigated more closely the data itself. They discovered that the data included numerous irregularities, such as differences in the TTL(time to live) for attacks versus normal traffic, that even a simplistic IDS could identify and achieve better performance than would ever be achieved in the real world [11]. Despite the warnings about the DARPA IDS evaluation dataset, Caswell and Roesch[12], and Brugger and Chow[11] found that the DARPA dataset may be still useful to evaluate the true positive performance of an ordinary network based IDS. According to Chan and Malhony's[10] work, it is apparent that, if an advanced IDS is unable to perform well on the DARPA dataset, it would not perform acceptably on realistic data.

In our research, we were not being able to get any publicly available source of real intrusion alert data with proper ground truth of attack scenarios. So, even though the DARPA data set is generally not regarded as suitable for testing intrusion detection systems, it was judged to be suitable for our use to test IDS alarm filtering.

5.2 Future Work

For the future work, we can consider the following issues:

- 1) to test with real time data, with updating of IDS configuration and network
- 2) to use multiple sensors rather than using a single sensor, both host based and network based
- 3) to test modeling with other observed patterns, than that of the alert types and, the relation to the source or destination of an attack.
- 4) to investigate whether the false alarms generated from the filter is really false alarms or real intrusive alarms
- 5) to investigate raw network data so that ISC algorithm can be tested whether it can behave like an anomaly based IDS
- 6) to test the ISC algorithm with different modeling for alert filtering purpose, for example, taking 10 or 20 alarm messages at a time rather than taking all messages in a certain time window.

6 References

1. Kwok Ho Law and Lam For Kwok. IDS False Alarm Filtering Using KNN Classifier. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005(pp. 114-121). ISBN: 978-3-540-24015-0
2. Pietraszek, T. Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection. Recent Advances in Intrusion Detection: 7th International Symposium RAID 2004(pp. 102-124), September 2004.
3. Ekman, Jan. Holst, Anders. Incremental Stream Clustering and Anomaly detection. SICS Technical Report T2008:1(31st January2008). ISSN 1100-3154.
4. Holst, Anders. Ekman, Jan. Larsen, Stefan. Abnormality Detection in Event Data and Condition Counters on Regina Trains. The Institution of Engineering and Technology International Conference on Railway Condition Monitoring(29-30 November, 2006), Page(s):53 – 56, Birmingham. ISBN: 0-86341-732-9, INSPEC Accession Number: 9309358. .
5. Lincoln Laboratory Massachusetts Institute of Technology. DARPA Intrusion detection evaluation.
<http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html>.
Accessed last on November 21, 2008 13:41 GMT.
6. Julisch, K. using Root Cause Analysis to Handle Intrusion Detection Alarms. PHD thesis, University of Dortmund(2003).
7. Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic; accessed last on 12/12/2008 23.50 GMT
8. Source: <http://www.snort.org> ; accessed last on 12/12/2008 23.55 GMT

9. McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. Information System Security 3 (4), (pp. 262-294).
10. Mahoney, M. V., P. K. Chan (2003). An analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for network anomaly detection. Proceedings of 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003), Volume 2820 of Lecture Notes in Computer Science, Pittsburgh, PA(pp. 220-237), Springer.
11. S Terry Brugger & Jedadiah Chow. An Assessment of the DARPA IDS Evaluation Dataset Using Snort(November 8, 2005). UCRL-CONF-214731.
<http://www.cs.ucdavis.edu/research/tech-reports/2007/CSE-2007-1.pdf> accessed last on 12/12/2008 22.55 GMT
12. Caswell, B., M. Roesch (16 May 2004). Snort: The open source network intrusion detection system. <http://www.snort.org/>. accessed last on 12/12/2008 23.57 GMT
13. Ron Gula(December,2002). Correlating IDS Alerts with Vulnerability Information. white paper, Tenable Network Security Inc.
Source: <http://www.nessus.org/whitepapers/va-ids.pdf> accessed last on 11/12/2008 13.47 GMT
14. Source: http://en.wikipedia.org/wiki/Type_I_and_type_II_errors accessed last on 11/12/2008 20.57 GMT
15. Christopher Kruegel, Fredrik Valeur, Giovanni. Intrusion Detection and Correlation, Challenges and Solutions. Book on Advances in Information Security(Volume 14). ISBN:0-387-23398-9. Springer.

16. Julisch, K.. Mining Alarm Clusters to Improve Alarm Handling Efficiency. Proceedings of 17th Annual Computer Security Applications Conference, December 2001(pp. 12-21).
17. Julisch, K. Clustering Intrusion Detection Alarms to Support Root Cause Analysis. ACM Transactions on Information and System Security, 6(4), November 2003(pp. 443-471).
18. Alshammari, R., Sonamthiang, S., Teimouri, M., Riordan, D. Using Neuro-Fuzzy Approach to reduce False Positive Alerts. Fifth Annual Conference on Communication Networks and Services Research(CNSR'07),14-17 May 2007(pp. 345 - 349) IEEE
19. Bloedorn, E., Hill, B., Christiansen, A., Skorupka, C., Talbot, L., Tivel, J. Data Mining for Improving Intrusion Detection. Technical report, MITRE(2000).
20. Source: http://en.wikipedia.org/wiki/Intrusion_detection_system accessed last on 11/05/2008 12.57 GMT
21. Source: www.ossec.net accessed last on 18/01/2009 13.47 GMT
22. Source: http://en.wikipedia.org/wiki/Data_mining accessed last on 29/12/2008 12.07 GMT
23. Source:
http://library.ahima.org/xpedio/groups/public/documents/ahima/bok1_025042.hcsp?dDocName=bok1_025042 accessed last on 18/01/2009 15.50 GMT
24. Source: http://en.wikipedia.org/wiki/Machine_learning_algorithm accessed last on 29/12/2008 12.30 GMT
25. Henry Liberman. Interaction is the key to machine learning applications. Workshop paper, Media Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., USA.
Source:<http://web.media.mit.edu/~lieber/Lieberary/AI/Interaction-Is/Interaction-Is.html>
accessed last on 29/12/2008 12.50 GMT

26. Source: http://en.wikipedia.org/wiki/Supervised_learning accessed last on 29/12/2008 12.35 GMT
27. Source: http://en.wikipedia.org/wiki/Unsupervised_learning accessed last on 29/12/2008 12.40 GMT
28. Source: http://en.wikipedia.org/wiki/Data_clustering accessed last on 29/12/2008 14.42 GMT
29. Source: http://en.wikipedia.org/wiki/Artificial_neural_network accessed last on 29/12/2008 15.12 GMT
30. Source: <ftp://ftp.cerias.purdue.edu/pub/papers/sandeep-kumar/kumar-intdet-phddiss.pdf> accessed last on 29/12/2008 15.12 GMT
31. Source: http://en.wikipedia.org/wiki/Precision_and_recall accessed last on 05/01/2009 11.42 GMT
32. Schoderbek, P. P. , Schoderbek, C. G., Kefalas, A. G. (1990). Management Systems: conceptual considerations(4th ed.). Homewood, IL : Business Publ.

7 Appendix A Abbreviations

IDS : Intrusion detection System

NIDS: Network based Intrusion detection System

HIDS: Host based Intrusion detection System

KNN : K-Nearest Neighbor

ISC : Incremental Stream Clustering

DARPA : Defense Advanced Research Projects Agency

JNI : Java Native Interface

TTL : Time To Live