Efficient Data Mining Algorithms for Intrusion Detection

Submitted By

Sanjay Rawat

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer & Information Sciences
University of Hyderabad
Hyderabad
June 2005

Efficient Data Mining Algorithms for Intrusion Detection

Approved by:

Professor Arun K Pujari, Adviser

Professor V. P. Gulati, Advisor

Head of the Department

Dean

Date Approved

To My Mother,

Who is no longer with me,

But must be watching me

And becoming happy for me to have accomplished this work.

I Love You, Mom

SELF-DECLARATION

I, Sanjay Rawat, hereby declare that the work presented in this thesis has been carried out by me under the guidance of Prof. Arun K. Pujari, Department of Computer and Information Sciences, University of Hyderabad and Dr. V. P. Gulati, as Director, IDRBT Hyderabad (currently associated with TCS, Hyderabad as Consulting Advisor), as per the PhD ordinances of the university. I declare, to the best of my knowledge, that no part of this thesis has been submitted for the award of a research degree of any other university.

(Sanjay Rawat)

ACKNOWLEDGEMENTS

There are many people whom I would like to thank for their support throughout my research program, but two stand out as the most important. These are my PhD supervisors Prof. Arun K. Pujari and Prof. V. P. Gulati. My terrific supervisor Prof. Pujari has taught me to be a good and effective researcher by imparting advice on all technical issues. During the initial days of my research, I was a bit scared of his very strict and demanding nature, but now I realize the importance of that kind of training. The best thing, apart from many others, that I could learn from him was the importance of an effective experimentation. Throughout my research, he has been very particular about experimentation. My second supervisor Prof. Gulati, apart from scanning my work and progress, has been very helpful in providing me very good infrastructure and resources. He has always shown confidence in me. In spite of his very busy schedule, he spared time for discussion and encouragement. In all, both of my supervisors contributed a lot in building my confidence as a good researcher.

I would like to thank faculty of IDRBT - Ms. V. Radha, Dr. Ashutosh Saxena and Dr. P. Radhakrishna for their useful discussion and advice throughout my research. I would also like to thank IDRBT for its fellowship, without which it would have been very difficult to pursue this research.

I would like to pay my gratitude to Prof. V. Rao Vemuri and his PhD student Liao, from University of California, Davis, for their help during my initial days of research. It has been a very good experience working with Prof. Vemuri for about three months.

I am thankful to my colleagues and friends - Ram, Nitin, Anshuman, Manila, Priyavrat, Haimesh, Atul, Lallu, Banti, Manik Da, John, Sandhya, Shalini, P. Sridhar, Pradeep, Zakir, Ravi, Mohan and many more, who have been a continuous source of entertainment and have made my days memorable. Directly or indirectly, they have contributed in the completion of this work. My special thanks goes to Dr. Challa S. Sastry, with whom I worked on some part of my research work. He has been very friendly and informative.

I am deeply indebted to the love and support of my family. This thesis is dedicated to the memory of my mother. She never considered me as a grown up boy, but as her still small kid and showered all her affection and care. My father has always supported me and never imposed anything on me, but at the same time, taught me the importance of being disciplined and punctual. I am proud of having them as my parents. I am also grateful to my elder brother and sister, sister-in-law and brother-in-law for their care and support. I also take opportunity to express my thank and love to my two nieces and two nephews. It's always fun in talking to them. I used to forget my tension by hearing their innocent voices. May God bless them with eternal happiness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The present thesis investigates some algorithms for intrusion detection. The focus of the thesis is on host-based anomaly system. However, we also propose network-based methods for IDS. For host-based systems, we analyze the system calls, invoked by the processes and for network-based systems, we analyze network packets.

We observe that while profiling a process for intrusion detection, not all of the system calls, invoked by the process, are important for behavior learning. We, therefore, investigate the applicability of Spectral Analysis technique - *singular value decomposition* (SVD) as a preprocessing step to reduce the noise (*not-so-important* system calls) of the data. This reduction highlights the most prominent features in the data by removing the noise. This preprocessing step not only makes the data noise-free, but also reduces the dimensionality of the data, thereby minimizing computational time.

In order to capture the similarity between two processes, we introduce a new similarity measure, termed Binary Weighted Cosine (BWC) metric, for anomaly-based intrusion detection schemes that rely on using sequences of system calls. The new similarity measure considers both the number of shared system calls between two processes as well as frequencies of those calls. The k nearest neighbor (kNN) classifier is used to categorize a process as either normal or abnormal. The proposed BWC metric enhances the capabilities of simple kNN classifier significantly especially in the context of intrusion detection. We also propose the use of Kendall Tau distance to capture the local ordering of system calls to enhance the proposed BWC metric.

We observe that the intrusive behavior of a process is highly localized characteristics of the process. There are certain smaller episodes in a process that make the process intrusive in an otherwise normal stream. As a result, it is unnecessary to consider the whole process in totality to characterize its abnormal behavior. We, therefore, make use of rough set theory to establish that subsequences of reasonably small length of sequence of system calls would suffice to identify abnormality in a process. Rough set theory also facilitates identifying rules for intrusion detection, which are very easy to interpret.

Inspired by the work on network traffic analysis, we investigate the usability of wavelets and SVD to identify the anomalies (DoS attacks). We propose an improved method to estimate Hurst parameter to determine the presence of self-similarity. Self-similarity is used as the signature for normal traffic. We further modify our method to locate the location of anomaly in the network data by combining SVD with wavelet-based approach.

All of the proposed methods are tested on standard data sets like DARPA system call data, UNM data, KDDcup'99 network data and the data collected from a live network. The empirical results show that the proposed methods perform better, when compared with similar existing methods.

CHAPTER I

INTRUSION DETECTION SYSTEMS: INTRODUCTION

If a man will begin with certainties, he will end in doubts; but if he will be content to begin with doubts, he will end in certainties.

-Francis Bacon (1561-1626), Advancement of Learning

In this introductory chapter, we describe *intrusion detection systems* (IDS) and some common approaches to build an IDS in section 1.1. The problem statement and the thesis contributions are described in section 1.2, which is followed by the description of the organization of the thesis in section 1.3.

1.1 Intrusion Detection Systems

In the modern network, IDS has become an important and integral part of over-all security architecture. In order to define an IDS, it is important to understand "what is an *intrusion*" and then "what is an *intrusion detection*." We take terminology and definitions from the NIST report [10].

An intrusion can be characterized in terms of *confidentiality, integrity, and availability*. An event or action causes breach of confidentiality if it allows to access resources, residing in a computer in an unauthorized manner. An event or action causes breach of integrity if it allows to change the states of resources, residing in a computer in an unauthorized manner. Similarly, an event or action causes breach of availability if it prohibits legitimate users to access resources or services, residing in a computer. *Intrusion detection* is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions. An *intrusion detection system* is a *software* or *hardware* that automates the process of monitoring and analyzing of events.

With the rapid growth of attacks, several intrusion detection systems have been proposed in the literature. Though the proposed systems differ from each other in some or many aspects, there are some basic components that are present in almost all the proposed systems. Figure 1 depicts a very simple generic architecture of a *typical* IDS [9]. In the

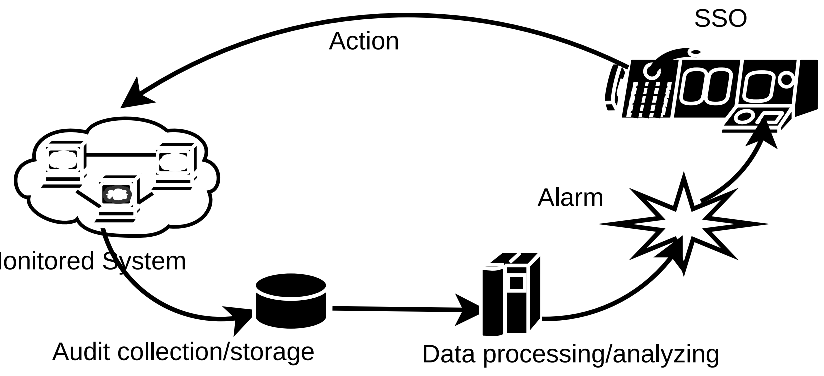

Figure 1: A generic architecture of IDS

figure, *monitored system* is the identity being protected. It can be a single host or whole network. *Audit collection/storage* collects the data to find events and processed them to put in proper format. *Processing unit* is the heart and mind of IDS. It is here where all the algorithms are executed to find the evidence of suspicious behavior. Upon detecting some intrusive behavior, an alarm is set off. Based on the capability of IDS, an action can be taken by the IDS to alleviate the problem itself. The alarm is also sent to the *site security officer*, who is in-charge of taking action against the attack.

1.1.1 Types of Intrusion Detection Systems

Intrusion detection systems can be categorized into various classes based on the components depicted in figure 1.

1.1.1.1 Network/Host Based IDS

Based on the audit collection/storage unit, there are two types of IDS. *Network-based* IDS (NIDS) collects data directly from the network that is being monitored, in the form of packets. Therefore, any NIDS is essentially a *sniffer*. Most NIDS are OS-Independent and are, therefore, easy to deploy. They provide better security against *DoS* attacks. But this type of IDS cannot scan protocols or content if network traffic is encrypted. Also intrusion detection becomes more difficult on modern switched networks, as packets are not accessible to

NIDS.

Under the same criterion, another type is *host-based* IDS (HIDS). HIDS collects data from the host, which is being protected, in the form of OS log files, system calls, CPU utilization, NT event logs, application level logs etc. These systems are ineffective by encrypted traffic or switched networks. However, HIDS are OS-dependent and thus require some prior planning before implementation. These systems are very efficient in detecting *buffer overflow* attacks.

1.1.1.2 Misuse/Anomaly Based IDS

Another criterion for categorizing IDS is from processing/detection standpoint. Based on *detection technique*, there are again two types of IDS. Misuse-based, also known as *signature-based*, IDS maintains a database of signatures of known attacks. Upon receiving data from the *audit unit*, it matches the data against the database and if any match if found, an alarm is triggered. For misuse-based detection, creating/representing the signatures is a challenging task and most of the research focus on this issue. It is obvious that this type of IDS are unable to detect *zero-day* attacks, as the signatures for these attacks are not available in its database. But the best thing about this type of IDs is that the false alarm rate is very low. Most of the commercial IDSs are from this category.

The second type of class is anomaly-based IDS, also known as behavior-based systems. Instead of keeping the signatures of known attacks, these systems learn the normal behavior of the entity, being monitored i.e. it keeps the signatures of the normal behavior. Any deviation from the normal behavior is considered as suspicious and an alarm is set off. Such systems work on the assumption that any abnormal behavior or activity differs significantly from the normal behavior. By definition, these systems are capable of detecting *zero-day* attacks. But these systems suffer from a high rate of false alarms because any deviation from normal activity may not be an intrusion. Therefore, reducing the false alarms is the focus of research under these systems.

There are some other criteria, too, to classify IDS into different categories. For example, based on *response*, an IDS can be *passive* or *active*. More details can be found in [9].

1.1.2 Various Detection Approaches

Building an effective IDS is an enormous knowledge engineering task. System builders rely in their intuition and experience to select the statistical measure for anomaly detection [129]. The basic task in IDS is to *learn* "what is normal" and "what is abnormal or an attack" and represent this knowledge to further alleviate security related problems. From this point of view, techniques from various disciplines have have been applied to build efficient systems. Tables 1 and 2 summarize some common approaches applied to misuse and anomaly based systems respectively [69].

Approach	**Description**
Expert systems	The knowledge about past intrusions, known system vulnerabilities and the security policy is encoded. As information is gathered, the expert system determines whether any rules have been satisfied. Such systems are characterized by their expert system properties that fire rules when audit records or system status information begin to indicate suspicious activity.
Keystroke Monitoring	The keystrokes entered by a user and the response of the computer during an interactive session are monitored and recorded.
Model based	In this approach, known intrusion attempts are modelled as sequences of user behavior. These behaviors are then modelled as events in an audit trail.
State transition analysis	The monitored computer system can be represented as a states transition diagram which is a graphical representation of the actions performed by an intruder to compromise system. An intrusion is viewed as a sequence of actions performed by an intruder that leads from some initial state on a computer system to a target compromised state. State transition analysis diagrams identify the requirements and the compromised state of the penetration. They also list the key actions that have to occur for the successful completion os an attack.
Pattern matching	It is the encoding of known intrusion signatures as patterns, to be matched against the audit data. It attempts to match incoming events to patterns representing intrusion scenarios. This model is based on the notion of an event, which consists of monitored changes in the state of the system, or part of the system. It can represent a single action by a specific user on a system, or an action by the system, or it can represent a series of actions resulting in a single, observable record.

Table 1: Approaches to misuse detection

1.2 *Problem Statement and Contribution*

The thesis mainly targets the anomaly-based intrusion detection systems and, therefore, it makes main contribution in the arena of anomaly-based IDS.

Approach	Description
Statistical	The normal or expected behavior is defined by collecting data related to the behavior of legitimate users over a period of time. Statistical tests are applied to observe behavior to determine behavior legitimacy. Statistical tests include threshold measures, like number of failed login attempts; mean and standard deviation of user profiles; multivariate models, which count correlation between multiple events; markov process model, which maintains a transition matrix of event's probabilities; clustering analysis to detect outliers.
Predictive pattern generation	Future events are predicted based on the events that have already occurred. Rules generated by the IDS define the probability that a certain event will occur. A rule consist of a left-hand side, defining two concurrent events, and a right-hand side, providing the probability of a specific event following the events defined on the left-hand side of the rule. An event is flagged as intrusive if the left-side of a rule matched, but right-hand side is statistically deviant from the prediction.
Neural networks	These systems learn to predict the next command based on previous commands by a specific user. Building a neural network based IDS involves following phases: (a) representing the data (e.g. user's command, system calls or network data) in vector form; (b) train the ANN to identify characteristic of data (e.g. user's identity or process execution path); (c) calculate the difference between ANN' output and actual event. If difference is large, then anomaly has occurred.
Sequence matching and learning	Lane and Brodley introduced the application of machine learning to IDS. This approach is based on the hypothesis that a user responds in a predictable manner to similar situations, which leads to a repeated sequence of actions. Thus a user can be profiled in terms of characteristic sequences of commands. A suitable distance metric is applied to compare two sequences. If the difference is greater than some threshold, the sequences, in question, do not represent the same user and hence, identifies a masquerader. Some immunological based approaches also fall in this category.

Table 2: Approaches to anomaly detection

1.2.1 Problem Statement

It is well known that anomaly-based IDS suffers from the high rate of false alarms. Continuous efforts are being made to reduce the high false positive rate. We believe that intrusion detection is a data analysis process and can be studied as a problem of classifying data correctly. From this standpoint, it can also be observed that any classification scheme is as good as the data presented to it as input. More clean the data, higher accurate results are likely to be obtained. From anomaly-based IDS point of view, it implies that if we can extract features that demarcate normal data from abnormal one properly, false positive rate can be reduced to a great extent. Therefore, in this work, we investigate the techniques which facilitate in the process of demarcating normal data from abnormal ones.

On the similar lines, we observe that most of the data mining and machine learning based methods in intrusion detection make use of well known tools and techniques. It may turn out that these general techniques are not very effective in classifying data as normal or abnormal with very high accuracy. There is a need to customize those techniques according to the requirement of intrusion detection. We also focus on this aspect in our work.

Apart from the problems mentioned above, the fast detection of attacks remains one of the focal points to be worried about. With the present complexity and variety of attacks, we need a huge amount of data to analyse and produce results. But larger the amount of data, longer the time to analyse it, which delays the detection of attacks. An IDS will be of more use if it can trigger an alarm early enough to reduce the damage that an ongoing attack can do. Thus, there is a need to make IDS as fast as to operate on-line. We believe that this can be achieved if we can reduce the data, to be analysed, without degrading its quality.

In a nutshell, the present thesis tries to answer the following problem:

> *How to make the IDS fast enough to process data on-line and detect attacks early and in case of anomaly-based IDS, how to reduce the false positives to an accepted level, with a high detection rate.*

1.2.2 Thesis Contributions

Though the research presented in this thesis mainly contributes to the area of IDS, it can also be used for other applications related to data mining. The main contributions of our thesis are as follows:

1. **Data Reduction for Fast IDS:** Most of the *system calls* based algorithms consider all the system calls invoked by the process, while analyzing the process. We observe that not all the system calls are important to study the behavior of a process. We, therefore, propose the use of *singular value decomposition* to remove such system calls, by considering them as noise. We provide a direct empirical proof for our observation. This reduction makes the data smaller and, therefore, leads to a fast IDS.

2. **New Similarity Metric:** We study the processes carefully in terms of system calls and find that general purpose similarity metric, viz. *cosine metric*, is not sufficient enough to capture the similarity among processes. We propose a new similarity metric, named as *binary weighted cosine* (BWC) metric. BWC is based on the frequency and number of common system calls between the processes. The results, obtained on DARPA dataset with kNN classifier, are far better than kNN classifier with cosine metric and comparable to recently proposed SVM classifier. Apart from introducing new similarity metric, our study also provides a direct empirical evidence that choice of proper similarity metric is of paramount importance, even with the well used classifiers. The BWC metric can be used in other domains like text mining as well.

3. **Pattern Extraction for On-line Detection:** Sequence matching in process profiling in terms of subsequences of system calls is a well known technique for intrusion detection. We observe that not all the subsequences of a process are responsible for anomaly detection. Anomaly, as a sequence of system calls, is localized in nature. Therefore, there are many subsequences which are common in both normal and abnormal processes. This phenomenon makes detection hard by raising false positives. In order to extract meaningful subsequences, we map IDS on a decision table and apply rough set based techniques to form rules. With this techniques, we do not have to wait for whole of the process to terminate first for analysis. We can analyse small sequences of system calls, while process is in running state. In this way, we propose an on-line method to detect intrusive behavior.

4. **On-line Analysis of Network Data:** The phenomenon of self-similarity is exhibited by network traffic. The absence of self-similarity can be attributed to some anomaly in the data. The presence or absence of self-similarity is calculated by estimating Hurst parameter. To estimate Hurst parameter, among other approaches, *energy-scale* plot is widely used. But the exact location of the anomaly and scale over which it is visible in the data is not detected properly. We propose a new method to calculate

Hurst parameter, which takes care of location and scale properly. We also provide theoretical observations, where *energy-scale* based approach may produce erroneous results. We extend our approach to determine the nature of anomaly i.e. a *short-term anomaly* or a *long-term anomaly*. Such identification helps us to generate early warning in case of a *long-term anomaly*.

5. Extensive Experimentation: Apart from introducing new techniques, we provide rigorous experimentation on well cited datasets, like DARPA'98, KDDcup'99 and UNM system calls data. Most of the results are accompanied by ROC curves and AUC scores. We also provide results on cross validation technique, leave-k-out, to avoid bias in the experimentations. Such empirical proofs approve the feasibility of the proposed methods in the thesis.

1.3 Thesis Outline

The thesis is organized as follows:

Chapter I: The first chapter, ***"Intrusion Detection Systems: Introduction"***, provides a general description of IDS and general techniques for intrusion detection. We also provide *problem statement* and the *main contributions* of our work, which is followed by the *organization of the thesis*.

Chapter II: In this chapter, entitled ***"Intrusion Detection Systems: A Survey of Algorithms and Approaches"***, we provide a survey on the existing techniques and methods on intrusion detection. The chapter covers most of methods and algorithms in an elaborated manner. The chapter covers the research on all aspects of IDS, from host-based to network-based and from anomaly-based to misuse-based. We also provide a brief introduction to available datasets to evaluate IDS. The chapter ends with some of the open problems in the area of IDS.

Chapter III: Entilted ***"Application of Singular Value Decomposition to Fast Intrusion Detection"***, the chapter provides the details of our work on the usability of *singular value* *decomposition* [67], to reduce the dimension of the data. The idea is inspired by a technique called *latent semantic indexing* [41]. The method makes use of kNN classifier to detect the attacks. The process is converted into the vector by calculating the frequencies of system calls. In this way, we get an *incidence matrix* from all the normal processes, which is decomposed by using SVD. We project each vector into a space of lower dimension by choosing few largest singular values. Therefore, each vector is reduced to some lower dimension and we apply kNN to further classify it. We have given empirical results to justify the use of SVD to reduce the dimension of the data. We show that this reduction does not degrade the accuracy of the classifier, by performing experimentation of DARPA dataset.

Chapter IV: From this chapter onwards, entitled ***"An Improved Similarity Metric for kNN Based Anomaly Detector"*** , our major contributions are described. The chapter presents cases where cosine metric based kNN scheme [120] produces result which may lead to erroneous conclusions about the similarity of the processes. We, therefore, propose a new similarity measure, termed as *binary weighted cosine* (BWC) metric. BWC is based on the frequency and number of common system calls between the processes. In doing so, each process is converted into the two vectors - one captures the frequencies of system calls while other captures the occurrence of system calls. BWC metric is used to calculate the similarity and kNN is used to classify the process as normal or abnormal. This scheme represents an example of *distance weighted* kNN classifier. We extend the above scheme by including the partial information about the order of occurrence of individual system calls in the process. For this purpose, we make use of *Kendall Tau distance*, which is used in rank aggregation [51]. We provide experimental results for each of the schemes on DARPA'98 data. The results are compared with relevant scheme, proposed in the literature.

Chapter V: The main motivation for chapter, entitled ***"Intrusion Detection Using Rough Set Theory"***, is the observation that in all the schemes, described in the earlier chapters, any similarity operations are performed on the process after its termination.

This may delay the detection. Also, not whole of the process is abnormal, as compared to normal process. Only a small part (or parts) of an abnormal process is abnormal, while most of it is similar to normal process. We, therefore, project an IDS as a decision table and apply *rough set* [151] based techniques to extract the feature for normal and abnormal processes. The processes corresponding to some attacks and normal ones are represented in a decision table. The *lower approximation* is calculated to discard the overlapping (common) subsequences between normal and abnormal processes. In this way, we get the *positive region* of the given data. We apply a rough set based rule learning algorithm - LEM2 [72] to generate IF-THEN type rules. These rules are used to classify the processes as normal or abnormal. We experiment on DARPA'98 data, using RSES tool. The approach is suitable for on-line detection, for we can compare the running process, without waiting for it to terminate.

Chapter VI: In this chapter, entitled ***"Network Traffic Analysis and Wavelets for Intrusion Detection"***, we focus on network based IDS. The idea is inspired by the work described in [66][13]. The self-similarity, which is exhibited in network traffic, is taken as the characteristics of normal traffic. The self-similarity is characterized by estimating Hurst parameter H. We extend the *energy-scale* plot based scheme for estimating H, by enabling it to detect the locality of the anomaly and the scale on which the anomaly is exhibited. For this purpose, we use wavelet theory and definition of self-similarity. The proposed scheme performs well on KDDcup'99 data. We further provide the extension of the above mentioned scheme. The algorithm proposed herein integrates the wavelet transform with singular value decomposition (SVD) for the analysis of self-similar network traffic data. The algorithm uses the properties of the right and left singular matrices, obtained in SVD of a matrix of local energies of wavelet coefficients, to determine the scales over which the data have possibly normal behaviour and locations at which the data have possible anomalous behaviour. We concentrate more on the theoretical aspects of our work. To show applicability of our method, we have taken a very small known self-similar data. However, to justify our approach empirically, we apply it on real network data, captured from an operational financial network INFINET [84] and Kddcup'99 dataset [89].

Chapter VII: In this chapter, entitled ***"Conclusions"***, we summarize our work, presented in the thesis. The conclusions are drawn based on the observations and empirical results. We also provide some future directions, in which the research, described in the thesis, can be extended.

We provide a detailed bibliography on intrusion detection research and relevant areas in the end of the thesis.

CHAPTER II

INTRUSION DETECTION SYSTEMS: A SURVEY OF ALGORITHMS AND APPROACHES

In Science the credit goes to the man who convinces the world, not to the man to whom the idea first occurs.

- Sir Francis Darwin (1848-1925, English botanist, son of Charles Darwin)

2.1 Introduction

In this chapter, we present a detailed survey of relevant literature in IDS. The aim of this survey work is to present the current research trends from the point of view of different algorithms used in building IDS. Efforts have been made to present an up-to-date research information with details of approaches and algorithms that have been developed during the last decade of intrusion detection research. One of the excellent survey papers is by Axelsson [9] which covers the intrusion detection research from 1988 to 1998. In 1980, Anderson [8] formally introduced the concept of intrusion detection system by identifying various entities involved in any kind of attacks. But the research in IDS received an impetus by a pioneer work by Denning [42]. Since then, there has been a number of research proposals to build and enhance the performance of IDS. The main difficulty in presenting the survey is to come up with a good classification structure so that all the methods can be put in either of the categories. We find that it is almost infeasible to divide the set of total intrusion detections systems into disjoints sets. We, therefore, study the proposed algorithms under the themes which best categorize the methods, e.g statistical approaches, data mining approaches etc. In case, where it is not possible to assign a category to any method, we use the data-collection technique as the category of that method.

The rest of the chapter is organized as follows. Section 2.2 describes some of the widely used datasets for the evaluation of IDS, e.g. DARPA'98 data, UNM system calls data and

Kddcup'99 data. In section 2.3, we concentrate on the research, which makes use of system calls for IDS. Section 2.4 covers approaches which are based on neural networks on network data. The research, based on artificial immune system, is covered in the section 2.5. The focus of the section 2.6 is the research, based on data mining techniques, which includes *association rules*, *clustering* and statistical approaches. In section 2.7, we analyze the methods, which are based on traffic flow analysis. This includes the application of wavelets and other statistical techniques. We conclude this chapter by providing some open problems in section 2.8.

The experimental results on any proposed method are very crucial to analyze its applicability. To evaluate a IDS, there are few standard datasets available for researchers. The following section describes few of them.

2.2 *Data Sets for the Evaluation of Intrusion Detection Systems*

Errors using inadequate data are much less than those using no data at all.

- Charles Babbage (1792-1871)

Most of the current intrusion detection systems are based on learning techniques from various disciplines like data mining, machine learning etc. It is well known that a learning algorithm is as good as the data used for training. Therefore, the availability of a good data set for intrusion detection systems is of great importance. A good data set not only facilitates the researchers to evaluate their proposals, but also provides a common platform to compare their work with others. In the following paragraphs, we give an overview of some of the widely accepted and used data sets available on Internet.

2.2.1 DARPA Data Set

The DARPA evaluation data set has been made available by MIT Lincoln Laboratory under DARPA sponsorship [126]. To the date, there are three data sets available for evaluation, DARPA 98, 99 and 2000 [35]. Each recent data set contains new attacks. The approach to create data sets is to simulate normal and attack traffic on a private network using real hosts, live attacks and live background traffic. Custom software automata simulate hundreds of programmers, secretaries etc running common UNIX applications. The custom software allows a small number of actual hosts to appear as if they were 1000's of hosts with different IP addresses. This corpus was designed to evaluate both false alarm rates and detection rates of intrusion detection systems using many types of both known and new attacks embedded in a large amount of normal background traffic. The corpus was collected from a simulation network that was used to automatically generate realistic trafficincluding attempted attacks. The attacks that are included, can broadly be classified into the following categories– Probing (information gathering attacks), Denial-of-Service (DoS, deny legitimate requests to a system), User-to-Root (U2R, unauthorized access to local super-user or root), and Remote-to-Local (R2L, unauthorized local access from a remote machine). The DARPA'98 data set contains, in all, over 300 attacks in the 9 weeks of data collected for the evaluation. These 300 attacks were drawn from 32 different attack types and 7 different attack scenarios. The attack types covered the different classes of computer attacks and included older, well-known attacks, newer attacks that have recently been released to publicly available forums, and some novel attacks developed specifically for this evaluation. Network based data is provided in *tcmdump* files and host based data in *BSM audit log* files. In DARPA'99 data set, some additional attacks were included and *NT audit log* files were also provided.

In response to the DARPA'98 data set, McHugh wrote a rather scathing critique of the evaluation [139]. While he presents many good points on how an evaluation of IDSs should be performed, he also criticizes numerous shortcomings in the challenge without acknowledging how difficult addressing some of the issues is. For example, it is noted that the generated data was not validated against real traffic to ensure that it had the same rates of malicious traffic versus nonmalicious anomalous traffic that caused false positives. Doing so would, however, require more insight into real traffic than we can possibly obtain (in particular, intent), further, modelling of traffic at that scale is still an area with much research left to be done. Some of his more directly applicable feedback was used for the IDS challenge the following year- DARPA'99.

Recent work of William and Marlin [6] shows that DARPA'98 data set shows self-similarity, but within a certain interval of time i.e. from 8 am to 6 pm. The periodogram method is used to estimate the Hurst parameter H. The methodology involves the plotting of periodogram for each two-hour period of each attack-free day in the DARPA data. The hurst parameter is estimated from each plot by performing a least square fit to the lowest 10% of the frequencies to determine the behavior of the spectral energy as it approaches the origin. The importance of their study is the observation that other methods that use temporal distribution for their model should concentrate only on the data between 8 am to 6pm of each day.

2.2.2 KDDcup'99 Data Set

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 *The Fifth International Conference on Knowledge Discovery and Data Mining*. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. The data set is derived from the DARPA'98 network data. The data is presented in a processed form using the approach suggested by Lee *et al* [115]. Each network connection is described by 41 features, which give details about *basic TCP features* and *time and window based features*. A separate feature (42^{nd}) labels the connection as normal or either type of attacks. There are approximately 5 million training records and 0.5 million testing records. Both training and testing data subsets cover four major attack categories: Probing/Scanning, DoS, User-to-Root, and Remote-to-Local.

It may be noted that a large set of machine learning and pattern classification algorithms trained and tested on KDD intrusion detection data set failed to identify most of the user-to-root and remote-to-local attacks, as reported by many researchers in the literature. Motivated by this observation, Sabhnani and Serpen investigate further to expose the deficiencies and limitations of the KDD data set to argue that this data set should not be used to train pattern recognition or machine learning algorithms for misuse detection for these two attack categories. It is empirically proved that the KDD training and testing data subsets represent dissimilar target hypotheses for user-to-root and remote-tolocal attack categories. The methodologies, employed for analysis, consisted of switching the roles of original training and testing data subsets to develop a decision tree classifier, cross-validation on merged training and testing data subsets, and qualitative and comparative analysis of rules generated independently on training and testing data subsets through the C4.5 decision tree algorithm. When using testing data as training data, RC4.5 detects only 6.14% U2R attacks, as compared to 76.92% with original training set. With a five-fold cross validation, training and testing data is merged and divided into 5 sets. Any four sets are used for training and remaining one is used for testing. Using this experiment, the detection rate is enhanced to 90%. These results demonstrate that classifier models trained using any four folds acquire sufficient information to achieve high detection rates if the fifth fold is employed for testing. This clearly indicates that the original KDD training and testing data subsets represent dissimilar target hypotheses for the U2R and the R2L attack categories.

2.2.3 UNM Data Set

Forrest first proposed the idea of system call tracing for program behavior. She established the analogy between *human immune system* and intrusion detection system. In order to test the proposed idea, the team led by Forrest collects the system call data for various privileged unix programs [188]. Some of the normal data are *synthetic* and some are *live*. Synthetic traces are collected in production environments by running a prepared script. Live normal data are traces of programs collected during normal usage of a production computer system. Each trace is the list of system calls issued by a single process from the beginning of its execution to the end and is given in a separate file. Each trace file lists pairs of numbers, one pair per line. The first number in a pair is the PID of the executing process, and the second is a number representing the system call. The mapping between system call numbers and actual system call names is given in a separate file. However,

using this mapping limits the use of *any program-any attack* combination. One has to use the corresponding attack files only. The system call traces are available for the following programs:
`synthetic sendmail, synthetic ftp, synthetic lpr, live lpr, xlock, live named, login and ps, inetd, sendmail`.

2.3 Research Based on System Calls Traces

In order for a process to request resources from the operating system, it must use the system call interface. By monitoring this interface, much can be learned about the behavior of the process being monitored. In the following paragraphs, we describe schemes which are based on system calls.

1. *tide scheme*

Motivated by the functioning of human immune system, Forrest *et al* proposed the idea of building behavior-based intrusion detection system by creating a definition of normal execution of a process based on traces of system calls invoked by the process [60][59]. In this approach, called *time-delay embedding* (tide), short sequences of system calls are used to profile a process. It uses a sliding window algorithm to populate a *table* with the positional relationships between system calls. The term *table* refers to a compressed representation of normal for an application where the current system call is used to index the table and there are at most N rows, where N is the number of unique system calls used by the application, being profiled. If during the creation of the table, two explicit windows with the same current system calls are encountered, the rows are collapsed into one (the rows corresponding to wait and time system calls in table 3 are the examples). Forrest *et al* use a sliding window of size $k + 1$ to record which system calls succeed or precede each other at offsets 1 through k. This implementation is said to have a "forward" lookahead because while matching with the testing process, the current system call is used as the index to the table and anomalies are found by performing a pair wise comparison between the current system call and each system call that follows at offsets 1 through k. If the process, under consideration, has a matching system call in the table at each offset, then it is considered as matching; otherwise it is a mismatch. The number of mismatches are recorded as a percentage of total number of mismatches. For a sequence of length L and lookahead of k, there are $(L - k)$ system calls with k-size window, for the size of the sequence including the index system call is $k - 1$; therefore, there are a total of $L - (k + 1) = L - k$ sequences and each will have k positions to match. Thus the total mismatches are $(L - k)k$. Also each of k-positions may be different. Therefor, the maximum number of pairwise mismatches for a sequence of length L with a lookahead of k is:

$$k(L - k) + (k - 1) + (k - 2) + ... + 1 = k(L - \frac{(k + 1)}{2}) \tag{1}$$

Whenever the number of mismatches for a testing process crosses a predefined threshold, the process is flagged as anomalous. The following example illustrates the approach:
Consider the following trace with $k = 3$:

```
select,wait,wait,access,mmap,time,time,open,fstat,read
```

Sliding window yields the following sequences of length 4:

```
select,wait,wait,access
wait,wait,access,mmap
wait,access,mmap,time
access,mmap,time,time
mmap,time,time,ope
time,time,open,fstat
time,open,fstat,read
```

Forward lookahead table representation is shown in table 3, which comprises a definition of normal. Let the testing process be:

```
select,wait,wait,access,open,time
```

For k=3, we get following three windows:

```
select,wait,wait,access
wait,wait,access,open
```

current call	offset 1	offset 2	offset 3
select	wait	wait	access
wait	wait access	access mmap	mmap time
access	mmap	time	time
mmap	time	time	open
time	time open	open fsat	fstat read

Table 3: Table representing the normal database for forward lookahead.

```
wait,access,open,time
```

the second and the third sequences are rejected by comparing with table 3. Using the equation (2.3), total number of mismatch are 12. Therefore,there are approximately 17% mismatches. The empirical results, performed on `sendmail` program, are impressive enough to attract other researchers to adopt and adapt this approach further. The experimental result shows that a sequence length of 6 is adequate to discriminate normal and abnormal behavior.

2. *stide scheme*

Inspired by the above mentioned work, *tide* approach is extended by Hofmeyr *et al* [77] by using a technique called *sequence time-delay embedding* (stide). In this approach, the database of normal sequences is used to monitor the ongoing behavior of the processes invoked by the program, by calculating Hamming distance between two (sub)sequences i.e. the difference between two (sub)sequences i and j is indicated by the Hamming distance $d(i, j)$ between them. For each subsequence i for incoming sequence, minimal Hamming distance $d_{min}(i)$ between i and set of normal sequences is determined as follows:
$d_{min}(i) = min\{d(i, j) \ \forall \ normal \ sequence \ j\}$. d_{min} value represents the strength of the anomalous signal and is used to calculated anomaly score. Thus for the whole new sequence, anomalous signal strength is given by
$S_A = max\{d_{min}(i) \forall \ new \ subsequence \ i\}$.
This S_A value is further normalized over the sequence length L, to compare S_A values for

different values of L i.e. $\hat{S_A} = \frac{S_A}{L}$

An anomaly count is defined as the number of mismatches in a temporally local region, called *locality frame*. If the count (locality frame count) is greater than a predefined threshold, the sequence is flagged as anomalous. For experimentation, `sendmail, lpr` and `ftp` programs are traced. It has been observed that for $k < 6$, `decode` intrusion is not detected, which supports the observation made by Forrest *et al* about minimal sequence length of 6. It has been noted in [77] that sometime false positives are caused by the misconfiguration of the system which indicates that the system is not functioning properly. Therefore, in some cases, such events are not false positives.

3. *GS ANN scheme*

Motivated by the ability of artificial neural networks to generalize from the past behavior, Ghosh *et al* propose the use of ANN to reduce the false positive and false negative [64]. In their approach, a host-based anomaly and misuse method is presented. A feed-forward multilayer (backpropogation) network is used for training and classification. As ANN can take only numerical values, it creates a problem to feed system call sequences to it. This problem is solved by creating some exemplar strings which characterize normal behavior. Encoding of the system calls data is done by calculating the distance of each incoming sequence as the distance from these exemplar string. For example, if there are m exemplar strings, then the whole sequence is converted into a vector of length m, wherein each entry indicates the the distance from corresponding exemplar string. However, nothing has been said about the creation of these exemplar strings. Number of nodes in input layer equals number of exemplar strings. There is one hidden layer with nodes varying from $10, 15, ..., 40, 50, 60$. In output layer consists of one node with values 0 and 1, indicating normal or anomalous behavior. In order to capture the temporal location of anomalous strings, Leaky-Bucket algorithm [187] is used. Leaky-Bucket algorithm increments a counter once it detects an anomalous signal and drops the counter value by one after a fixed period of time. Once the level in leaky-bucket crosses a predefined value, the sequence is flagged as abnormal. This is based on the observation, also mentioned in

[190][77], that intrusive behavior is localized. In order to draw ROC curve, various leaky-rates are considered. The same approach is used in misuse-based method by learning the abnormal behavior. This ANN based approach achieves a detection rate (DR) of 77.3% with 2.2% false positives (FP) for anomaly-based method and 90% DR with 18.7% FP for misuse-based method, on BSM data from DARPA 98 data set [35].

4. *KS DFA scheme*

The use of deterministic finite automata (DFA) is suggested to capture the temporal ordering of events in a sequence of system calls [97]. A program structure is learnt by constructing DFA using *macros*, which are variable-length patterns of system calls. Repeated patterns of system calls can be represented efficiently and compactly using DFA. First, the traces are separated into three categories, identified by the authors for `sendmail` program. Each process is divided into three parts: a prefix, a main portion, and a suffix. for each category, the longest common prefix and suffix is found. The frequently occurring system calls and common short patterns of system calls in the main portion are derived. Using these system calls and prefix and suffix, macros are defined. For `sendmail` program, a total of 24 macros are calculated. Each process is now reduced to a new strings made of these macros. Using these macros, the behavior of a process is represented by constructing DFA. When applied on `sendmail`, 147 processes are reduced to 26 different process description. Each new process is matched with theses macros, using the *locality frame count*, defined in [77]. The process is declared as normal or abnormal on the basis of some threshold. The main advantage of this scheme is that this has resulted in shorter normal profile. But identifying all the possible states, to be used in DFA, is very difficult and time consuming. If a sequence enters a state, not represented in DFA, the DFA can not move further. In other words, DFA is not flexible enough to capture all the states. If, however, all the states are incorporated in DFA, then it becomes too general to identify anomalous behavior.

5. *GSS Elman ANN scheme*

To overcome the issues, observed in the above approach, Ghosh *et al* propose the use of Elman Networks [65]. Ghosh *et al* advocate the use of a system which maintains some degree of state between inputs i.e. a recurrent ANN topology. A recurrent topology is one in which cycles are formed by the connections. Thus a input can affect the state, and the state can affect the classification. They make use of Elman recurrent network for their approach. The process of data preparation and presentation remains the same as in [64]. Elman network is used to predict the next input, I_{n+1}, by seeing the current input I_n. If O_n is the output for n^{th} input I_n, then the difference $O_n - I_{n+1}$ is the anomaly score for the sequence. Leaky-bucket algorithm is used to capture the temporal nature of attacks. This approach, based on Elman network, outperformed the approaches based on ANN [64] and DFA[97]. It could achieve a 100% DR with a very low FP.

6. *CLM anomaly dictionary scheme*

All of the approaches, described above takes the sequence of system calls, invoked by the process under normal execution to form the normal profile of the processes. Cabrera *et al* investigates the idea of forming the anomaly profile or *anomaly dictionaries* of various attacks for `sendmail` traces [24]. The basis methodology is same as *stide* for system calls data processing. Two approaches are discussed based on misuse and anomaly based techniques i.e. a hybrid approach.

Scheme 1:

Under this scheme, sequences of normal and abnormal traces are collected for `sendmail` program. Four traces of attacks and 78 traces of normal execution are used for training. Testing set consists of 6 traces of attacks and 20 traces of normal processes. The *relative anomaly count* is calculated for abnormal and normal traces as α_i^A, $i = 1, 2, ..., 8$ and α_i^N, $i = 1, 2, ..., 78$ respectively. Relative anomaly count is defined as the total number of anomalous sequences found in the trace, divided by the total number of sequences in the trace.

Finally, anomaly Count for anomalous and normal traces are defined as:

$$\alpha^A = \frac{1}{S}\sum_{i=1}^{S} \alpha_i^A$$

$$\alpha^N = \frac{1}{T}\sum_{i=1}^{T} \alpha_i^N$$

In this case $S = 8$ and $T = 78$. A detection threshold t is calculated as $t = \frac{1}{2}(\alpha^A + \alpha^N)$. Traces in the testing set are labeled as anomalous if their relative anomalous count α is larger than t; otherwise normal.

Scheme 2:

In order to identify the type of anomaly, the concept of *anomaly dictionary* is introduced. Anomaly dictionary is constructed for each anomaly. The anomaly dictionary for a given *anomaly* is the set of anomalous sequences (i.e. the sequences not found in the dictionary of normal sequences) that appear in the available anomaly trace. For any incoming trace with anomaly count α, let $\alpha^{[l]}$, $l = 1, 2, ..., N$, denote the anomaly count for anomaly l. The classifier labels the incoming trace as belonging to class l^* if $\alpha^{[l^*]} > \alpha^{[l]}$, $\alpha^* \neq l$ for all $l = 1, 2, ..., N$. The trace is normal if $al^{[l]} = 0$ for all $l = 1, 2, ..., N$. It has been observed that there was no overlap between the dictionaries for abnormal traces and those of normal traces. When the length k of sequences is at least 6, `decode` and `syslog` attacks are correctly labelled, once again supporting the observation made in [60] regarding the minimal length of sequence.

7. *LSC RIPPER scheme*

All of the approaches, mentioned so far, use sequential string matching in one or other forms. Though these are simple to understand and easier to implement, these approaches may be time consuming in case of a very big data set. Lee *et al* [117] investigate the idea of applying rule-based approach to find the rules for normal and abnormal classes. For their approach, lee *et al* use a rule learner called RIPPER [32] to generate the rules. RIPPER is the improvement over a well known rule learner IREP. RIPPER produces rules for the 'minority' class which, in this case, is abnormal class. A default rule is produced which classifies rest of the data as belonging to normal class. Training data is derived from UNM `sendmail` traces. Using a sliding window, normal traces are divided into short sequences of system calls. Similarly abnormal sequences are also formed. Each sequence is labelled as either normal or abnormal. To calculate the locality frame count, on the whole output of RIPPER, a sliding window of length $2l + 1$, $l = 3, 4, ...$, is used. Within this region of length $2l + 1$, if more than l predictions are abnormal, then the current region is predicted as abnormal region. If the percentage of abnormal regions is above a threshold, then the trace is an intrusion. There will be total of $n - (2l + 1) + 1$ subsequence of length (2l+1) in the sequence of length n. On an average, 200 to 28 rules are produced for classification, which makes the classification time shorter as compared to Forrest's approach. It should be noted that in order to have a anomaly-based intrusion detection, we must have number of abnormal sequences more than that of normal sequences. In other set of experiments, Lee *et al* learn to predict the nth system call after seeing the first $(n - 1)$ system calls. This approach tries to learn the co-occurrence of system calls under normal execution of a process. RIPPER outputs the rules with a confidence value associated with them. The confidence value is calculated as the number of matched examples divided by the sum of matched and unmatched examples. Whenever a rule is violated, its score is incremented by 100 times its confidence. The average score of entire trace is used to decide whether an intrusion has occurred.

8. *LD information theoretic measures*

The basic assumption for anomaly detection is that there is intrinsic characteristic or regularity in audit data that is consistent with the normal behavior and thus distinct from the abnormal behavior. Various information-theoretic measures are discussed in [112]. We summarize them in following paragraphs:

Entropy:

For a dataset X where each data item belongs to a class i.e. $x \in C_X$, the Entropy of X relative to this $|C_X|$-wise classification is defined as

$$H(X) = \sum_{x \in C_X} P(x) log \frac{1}{P(x)}$$

where $P(x)$ is the probability of x in X. For anomaly detection, we can use Entropy as a measure of the regularity of audit data. The smaller the Entropy, the fewer the number of different records and thus, more regular is the audit data. Therefore, anomaly detection models, constructed using datasets of smaller Entropy are likely to be simpler and have better detection performance and low false positives.

Conditional Entropy:

The conditional entropy of X given Y is the entropy of the probability distribution $P(x|y)$, that is,

$$H(X|Y) = \sum_{x,y \in C_X, C_Y} P(x,y) log \frac{1}{P(x|y)}$$

$P(x,y)$ is the joint probability of x and y. For anomaly detection, we can use conditional Entropy as a measure of regularity of sequential dependencies. Smaller the conditional entropy, better is the model.

Relative Entropy:

The relative Entropy between two probability distributions $P(x)$ and $q(x)$ that are defined over the same $x \in C_X$ is

$$relEntropy(p|q) = \sum_{x \in C_X} P(x) log \frac{p(x)}{q(x)}$$

Training and testing datasets must have same or very similar regularity for anomaly detection model to attain high performance. Relative Entropy measures the distance of regularities between two datasets.

Information Gain:

When constructing a classifier, a classification algorithm needs to search for features with high information gain, which is the reduction of Entropy when the dataset is partitioned according to the feature values. The information gain of an attribute (feature) A on dataset X is

$$Gain(X,A) = H(X) - \sum_{v \in Values(A)} \frac{|X_v|}{|X|} H(X_v)$$

where $Values(A)$ is the set of possible values of A, and X_v is the subset of X where A has value v.

When we model a sequence, the smaller the conditional entropy, the higher the information gain, and hence the better detection performance of the model. The method of applying these measure is as follows. Let X represent the set of length-n sequences of system calls, and Y be the set of (prefix) subsequences of the length $n-1$, the conditional entropy $H(X|Y)$ then measures the regularity of how the first $n-1$ system calls determines the the nth system calls. In more details, for each unique $x \in X$, $|x|$ is the number of occurrences of x in X, and $y(x)$ be the length-$n-1$ subsequence of x, i.e. if $x = (s_1 s_2 \cdots s_{n-1} s_n)$, then $y(x) = (s_1 s_2 \cdots s_{n-2} s_{n-1})$, then $H(X|Y) = \sum_{x \in X} \frac{|x|}{|X|} log \frac{|x|}{|y(x)|}$. When the above measures are applied to UNM `sendmail` data, it is found that conditional entropy of the normal dataset drops to very small values after sequence length reaches 6 or 7, which, once again, confirms the observation made by Forrest *et al.* To put it all together, the usability of the aforementioned measures can be summarized in the following steps [112]:

- Measure the regularity of the data and perform the appropriate transformation. Iterate this process, if necessary, so that dataset used for modelling has high regularity.
- Determine how the model should be built, i.e. how to achieve the best or optimal performance/ cost trade-off, according to the regularity measure.
- Use the relative entropy measure to determine whether a model is suitable for a new dataset.

Often, such techniques can be used to ensure small generalization error.

9. *t-stide*

A simple addition to stide, called *stide with frequency threshold* (t-stide) is proposed to test the premise that rare sequences are suspicious [190]. For each sequence in database, frequency of its occurrence in training data is also recorded. Sequences from test traces are compared to those in the database, just like stide. Rare sequences, as well as those not included in the database, are counted as mismatches. These mismatched are aggregated into locality frame count. Some better results are found if the length of the sequences is not fixed [192].

10. *TC LERAD variants scheme*

All of the above approaches concentrate only on the sequences of system calls. Tandon and Chan [184] propose to consider system calls arguments and other parameters, along with the sequences of system calls. They make use of the variant of a rule learner LERAD (Learning Rules for Anomaly Detection), originally proposed in [131]. LERAD is a conditional rule-learning algorithm that selects good rules from a vast rule space.Three variants of LERAD are proposed to generate rules under different inputs.

S-LERAD: The sequences of system calls of length six are used as input to LERAD (i.e. current system calls and the previous five system calls). This variant of LERAD is expected to find correlations among system calls. A simple rule learnt by *S-LERAD* is represented as:

$$SC_1 = close(), SC_2 = mmap(), SC_6 = open() \Rightarrow SC_3 \in \{munmap()\} \ \ (455/1)$$

This rule can be interpreted as whenever first system call is `close`, second is `mmap` and sixth is `open`, 455 times the third system call is `munmap`. The number 1 in the denominator indicates the only value (i.e. munmap) as a consequent, for this rule.

A-LERAD: This variant of LERAD is based on the hypothesis that arguments to system calls and other key attribute information is integral to modeling a good host-based anomaly detection system. The system call is taken as *pivotal attribute*. Any value for other arguments (path, return value, error status), given a system call, which was never encountered in the value for a long time, would trigger an alarm. If there are no conditions in the antecedent, LERAD accepts any attribute in the consequent of the rule. A sample rule is shown below.

$$SC = munmap() \Rightarrow arg_1 \in \{0\text{x}134, 0102, 0\text{x}211, 0\text{x}124\} \ \ (500/4)$$

According to this rule, `munmap` system call is normal if it takes arguments given by consequents of the rule. This rule is satisfied by 500 instances and as arguments, there are 4 different values of `munmap` system call.

M-LERAD: The third variant of LERAD merges both S-LERAD and A-LERAD. Each input comprises of system call, arguments, path, return value, error status and the previous five system calls.

The records that match the antecedent but not the consequent of a rule are considered anomalous. The degree of anomaly is based on a probabilistic model. For each rule, from the training data, the probability p of observing a value not in the consequent is estimated by $p = Pr(X \notin \{x_1, x_2, ...\} \mid A = a, B = b, ...) = \frac{r}{n}$, where $r = |\{x_1, x_2, ...\}|$, in the consequent and n is the number of records, satisfying the antecedent. During detection phase, when a novel event is observed, the degree of anomaly is estimated by:

$AnomalyScore = \frac{1}{p} = \frac{n}{r}$

A time factor t is also introduced to capture the temporal nature (locality frame) of attacks. Since a record can deviate from consequents of more rules, the total anomaly score of a record is given by:

$TotalAnomalyScore = \sum_i \frac{t_i}{p_i} = \sum_i \frac{t_i n_i}{r_i}$

For experimentation, DARPA BSM data is used. Empirical results show that the proposed approaches perform as good as stide and t-stide. In case of R2L attacks, A-LERAD and M-LERAD perform much better than stide and t-stide. It is also observed that adding argument level information is sufficient to detect R2L attacks. Sequence correlation is not sufficient to detect attacks specially R2L attacks.

11. *"why 6?: an observation*

It should be noted that in all of the approaches, discussed above, a minimum length of six for system calls sequence has been preferred. A minimum sequence length of six produce best result with UMN `sendmail` data. This is true if the classifier is using sequential string matching approach e.g. stide. Out of curiosity, Tan and Maxion investigate the UMN `sendmail` data with stide approach to find out "why 6?" [182]. A careful examination of abnormal data set reveals that the `decode1 sm-280.int` attack contains a minimal foreign sequence (MFS) of length six. A minimal foreign sequence is a foreign sequence such that all of its proper subsequences are already exits in the set of normal sequences. Therefore, in order to detect this foreign sequence, the length of the detector window (sliding window) must be at least six. In [112], it is suggested that conditional entropy may be useful for selecting appropriate sequence lengths for probabilistic classifiers. In other words, regularity of data does affect the performance of a classifier. But as stide is blind to probability distribution, its performance remains same even for data of different regularities [182]. This study also makes it clear that detector window length is data dependent. If we test the stide on some other data, e.g. DARPA, it may happen that length six is not appropriate. A rigorous experimentation is performed in [182] to establish the “why 6?”. In another study by the same authors, Tan and Maxion, it is mentioned that difference in the data regularity affects the performance of the detector [138]. According to this study, conditional relative entropy affects the performance of anomaly detectors, as has been observed in [112]. However, experiments performed in [182], show that tide is not affected by conditional entropy of the data. Therefore, such observations make it clear that classifier which are blind towards probabilities of events, are not affected by the regularities of data.

12. *Undermining anomaly detector*

Security is like a game of chess: one must anticipate all the moves the attacker may make and make sure that system will remain secure against all the attacker's response. In order to subvert an anomaly-based detector, there are two approaches: either insert the attack in the normal data while training or make the attack that look like normal activity. From attacker viewpoint, it is difficult to follow first approach. The later approach is feasible but needs a careful examination of IDS algorithm and data being analyzed. In [183], a method for undermining the stide algorithm [77] is discussed. It is shown in [182] that stide can detect an attack if it contains at least one MFS which is, at most, as lengthy as the detector window of stide. Any MFS remains undetectedd if its length is greater than the length of the detector window. A comparison of the size of detector window with the ability to detect different sizes of minimal foreign sequence is shown in figure 2.

In order to show that almost any malicious sequence of system calls can be shifted into

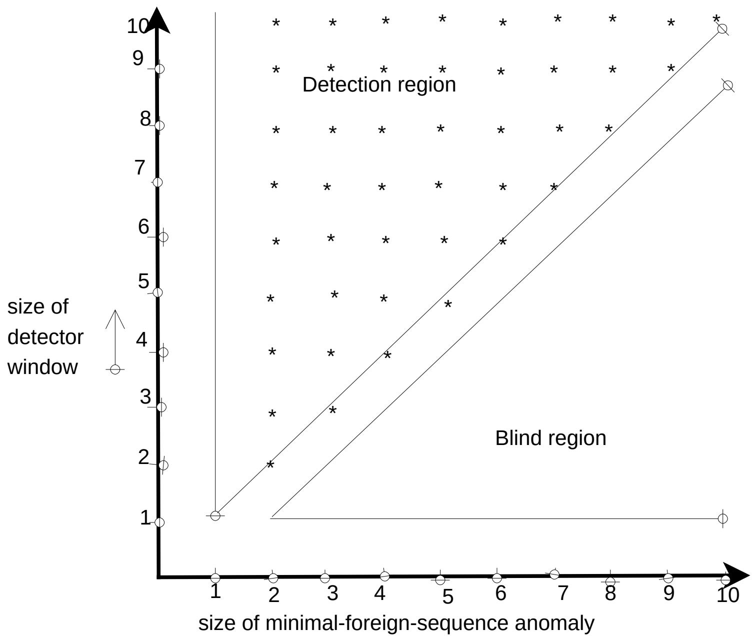

Figure 2: The detector coverage (detection map) for stide.

blind region of stide, two attacks viz. `passwd` and `traceroute` are chosen. `passwd` attack is attributed to the presence of the following subsequence of system calls: `setuid, setgid, execve` and `traceroute` to: `brk, brk, brk, setuid, setgid, execve`. Both of these attacks result in getting a shell with root privilege. It is also possible to get shell by executing the attack script such that it invokes only the valid system calls, which are present in the normal dataset. For example, for `passwd` attack, sequence `chmod, exit` may achieve the same goal. It should be noted that larger the length of MFS, more difficult it will be for stide to detect it. Thus, attacker should try to make MFS as lengthy as possible. This is quite feasible as stide does not consider arguments to system calls and therefore, system calls can be injected into the stream with 'no-op' option.

13. *CGSPS rough set anomaly scheme*

Cai *et al* [25] make use of rough set theory to induce rules for anomaly detection. Let X be the set of system calls sequences under the normal execution of a process. By using a sliding window of length $L+1$ along every sequence in X, a set of *normal subsequences* U is obtained. For every subsequences in U, the first L system calls are defined as *common* *positional attributes*. They constitute the set A of *common positional attributes*. The $(L+1)$th system call is defined as the *last positional attribute* and denoted by d. In rough set terminology, A is the set of *conditional attributes* and d is *decision attribute*. The basic idea of the *normal model* is the decision of the $(L+1)$th system call based on the preceding L system calls. A *normal model* $M(L, B)$ of a process is a set of prediction rules characterizing the relationships between A and d in the set U. The algorithm for anomaly detection based on normal model is summarized as follows.

I. Use a window of size $L+1$ to scan a sequence and generate a subsequence of length L.

II. Search for the rule in $M(L, B)$ matched by this subsequence.

III. If no rule is matched, most frequent prediction of the rules in $M(L, B)$ is used as the prediction.

IV. If there is more than one match, prediction is determined by voting.

V. If prediction accords with the $(L+1)$th system call of the subsequence, go to step 7. Otherwise go to step 1.

VI. If prediction is not the same as the $(L+1)$th system call, the prediction fails and the subsequence contradicts $M(L, B)$. If the entire sequence is scanned, go to step 7.Otherwise go to step1.

VII. Calculate the *anomaly index* and use a predefined threshold to determine if this process is normal.

The *anomaly index* of a system call sequence is defined to be the ratio of the number of its subsequence that contradict the normal model to the total number of subsequences in the sequence. `sendmail` dataset is used for experimentation. Only 2% of the normal data is used for training. Empirical results show that at length 20, 30 and 40, the proposed scheme performs better than Forrest *et al* scheme. The use of reducts further minimizes the number of attributes to be matched while calculating the anomaly index, which makes the decision faster.

14. *WDD variable-length patterns scheme*

Most of the approaches, based on system calls profiling based on fixed-length sequences. Wespi *et al* investigate the idea of variable-length sequences of audit trails [192][38]. Their approach is based on the observation that there are many cases in which very long sequences are repeated frequently. Also most of the sequences begin and end with similar subsequences. In such cases it is better to have a method for generating variable-length sequences. Wespi *et al* make use of the Teiresias algorithm [159], which is used to discover rigid patterns in unaligned biological sequences. The Teiresias algorithm is used to derived *maximal variable-length* patterns from the training data. A pattern p is maximal if there is no other pattern q that contains p as a subsequence and has same number of occurrences as p. Let there be a set of strings $S = \{s_1, s_2, \ldots, s_n\}$ over the alphabet $\sum = c_1, c_2, \ldots, c_n$.
A character $c \in s$ is said to be covered by the pattern p if $c \in p$ and p is a substring of s.
A string s is said to be covered by a set of patterns P if for each character c, $c \in s$, there is a pattern p, $p \in P$, such that c is covered by p.
The function $bCover(p, s)$ returns the number of characters covered at the begining and end of a sequence s by a pattern p.
The boundary coverage of a pattern p and a string set $S = s_1, s_2, \ldots, sn$, written as $bCover(p, S)$, is defined as

$$bCover(p, S) = \sum_{i=1}^{n} bCover(p, s_i)$$

An algorithm is devised to further reduce the set P of maximal variable-length sequences. The reduced pattern set R is constructed as follows. μ denotes the minimal pattern length that is used to generate the set of maximal variable-length patterns.

I. If $P = \phi$, then add all $s \in S$ to R and exit.

II. For each $p \in P$ calculate $bCover(p, S)$.

III. Select a pattern $r \in P$ for which $bCover(r, S)$ is maximal, i.e. there is no other pattern $p \in P$ for which holds:

- $bCover(p, S) > bCover(r, S)$ or,
- $bCover(p, S) = bCover(r, S) \wedge |p| > |r|$.

IV. Add r to R and remove it from P.

V. Remove all matching substrings adjacent to the beginning or end of a string, i.e., remove strings of the form $s = r^+$, and replace strings of the form $s = r^+s',\ |s'| > \mu$ or $s = s''r^+,\ |s''| > \mu$, with s' or s'', respectively.

VI. Remove the matching substrings that are not adjacent to the beginning or end of a string, i.e., as long as there is an $s \in S,\ s = s'rs'',\ |s'| \geq \mu,\ |s''| \geq \mu$, replace s with the two strings s' and s''.

VII. If there is an $s \in S$ with length $|s| < 2 \times \mu$, remove s from the set of strings S and add it to the pattern set P

VIII. If $S \neq \phi$, go to Step 1, otherwise exit.

In the pattern matching phase, The incoming stream is matched with the patterns in R. Whenever the number of unmatched system calls crosses a threshold, an alarm is raised. The pattern matching is done as follows:

I. Set the look-ahead parameter to a value $\delta > 0$, and set the threshold for the number of consecutive uncovered characters to a value $\tau > 0$.

II. Set the counter of consecutive uncovered characters, κ, to 0.

III. When there is a sufficient number of characters in the input stream I, find a pattern $p \in R$ that covers the beginning of the input stream I. If no pattern can be found, go to Step 6.

IV. Find $\delta > 0$ patterns $q_1, q_2, \ldots, q_\delta$, such that the string $t = pq_1q_2 \ldots q_\delta$ covers the beginning of the stream. If there are ϵ patterns $q_1, q_2, \ldots, q_\epsilon$, $0 < \epsilon < \delta$, that cover the entire input sequence, set $t = pq_1q_2 \ldots q_\epsilon$.

   - If t matches the entire input sequence, remove it and go to Step 2.

- If δ patterns can be found that cover the beginning of the input stream, remove pattern p from the input stream, and go to Step 2.

V. Determine all pattern combinations that match the beginning of the input stream. If there is a match, select the pattern combination that covers the longest input sequence, remove it from the input stream, and go to Step 2.

VI. Skip one character, and increase κ by 1.

VII. If $\kappa = \tau + 1$, raise an alarm.

VIII. Go to Step 2.

Using this approach, the size of database of normal pattern is reduced by almost 10 times. Experimental results are performed on *ftpd* data and comparison is made with *stide* approach.

15. *Eskin probability distribution scheme*

All of the above mentioned approaches assume the availability of clean data (labeled data), which in practice may not be available or difficult to get. Eskin *et al* present a method for detecting anomalies without training on normal data [53, 54]. The method makes use of statistics for explaining anomalies as outliers. The approach assumes that anomalies occur very rarely in the data with the probability λ against a large probability $1 - \lambda$ of normal data. An element x_i is either generated from majority distribution **N**, or with probability λ from minority distribution **A**. Then the generative distribution for the data is:

$$\mathbf{D} = (1 - \lambda)\mathbf{N} + \lambda\mathbf{A}$$

The approach makes an iterative method to partitioned the data into normal (N_t) and abnormal (A_t) data sets. Initially, $N_0 = D$ and $A_0 = \phi$. Any machine learning technique can be used to model probability distribution, P_{N_t} by using a function τ_N i.e.

$$P_{N_t}(X) = \tau_N(N_t)(X)$$
$$P_{A_t}(X) = \tau_A(A_t)(X)$$

The likelihood, L of the distribution $\mathbf{D}$ at time t is given by:

$$L_t(\mathbf{D}) = \prod_{i=1}^{n} P_D(x_i) = ((1-\lambda)^{|N_t|} \prod_{x_i \in N_t} P_{N_t}(x_i))(\lambda^{|A_t|} \prod_{x_j \in A_t} P_{A_t}(x_j))$$

For computational purpose, log likelihood (LL) is used as:

$$LL_t(\mathbf{D}) = |N_t| log(1-\lambda) + \sum_{x_i \in N_t} log(P_{N_t}(x_i)) + |A_t| log\lambda + \sum_{x_j \in A_t} log(P_{A_t}(x_j))$$

The likelihood of each element x_i for being outlier is measured by comparing the difference in the LL if the element is removed from N_{t-1} and in anomalous distribution A_{t-1}. If this difference $(LL_t - LL_{t-1})$ is greater than some threshold c, the element is declared as anomalous and permanently removed from N_{t-1}, otherwise it remains in the same set. This procedure is applied to all elements and finally the data is partitioned into two disjoint sets. Thus so created normal data set can be applied to learn the classifier. The proposed approach shows the better results in comparison to two well known methods viz. *stide* and *t-stide*.

16. *LV kNN scheme*

An approach based on *kNN classifier* is proposed by Liao and Vemuri [120] where the frequencies of system calls used by a program (process), rather than their temporal ordering, are used to define program's behavior. Their approach draws an analogy between text categorization and intrusion detection, such that each system call is treated as a word and a set of system calls generated by a process as a document. The processes under normal execution are collected from the DARPA data and thereafter converted into vectors, consisting of the frequencies of the system calls made by them during the normal execution. Let S (say, $Card(S) = m$) be a set of system calls made by all the processes under normal execution. From all the normal processes a matrix $A = [a_{ij}]$ is formed, where a_{ij} denotes the frequency of i^{th} system call in the j^{th} process. In order to categorize a new process P into either normal or abnormal class, the process P is first converted into a vector. The *kNN classifier* then compares it with all the processes A_j in A to determine the k nearest neighbors, by calculating the similarity $CosSim(P, A_j)$, using the cosine formula given by

$$CosSim(P, A_j) = \frac{P \cdot A_j}{\parallel P \parallel \cdot \parallel A_j \parallel} \tag{2}$$

The average similarity value of the k nearest neighbors is calculated and a threshold is set. When the average similarity value is above the threshold, process P is considered as normal; otherwise abnormal.

2.4 *Neural Network Based Approaches on Network Data*

Artificial neural networks (ANNs) have been one of the widely used techniques to classify the data, mainly due to their ability to generalize learning from their past experience.

17. *BPSS NN SOM scheme*

Bivens *et al* [20] propose a simple idea of detecting DoS attacks by applying NN and SOM. In their approach, first the N monitoring ports are decided and then number of packets per service port to victim machine, within a time period dt, is counted. According to the intensity of the traffic, the source IP addresses are clustered using SOM. The SOM contains a grid of neurons, each possessing a weight vector of length N. After training, the weight vectors reflect the number of hits to a particular port in time interval dt. Euclidean distance is used to decide BMU. To develop the clusters from SOM, a frequency value β to count how many times a particular neuron and its neighborhood were chosen as BMU, is calculated. The neurons with highest value of β are selected as centroids. β is calculated as follows:

$$\beta_{i,j} = freq_{i,j} + \sum_{x=1}^{reach} (\frac{freq_{i-x,j} + freq_{i,j-x} + freq_{i+x,j} + freq_{i,j+x}}{1+x})$$

where, $freq_{i,j}$ is the number of times the neuron at position(i, j) was BMU and $reach$ is the space we enforce between cluster centroids. The number of clusters (M) are given as input to the SOM. The NN is constructed as having $N \times M$ input nodes. The network is trained on limited part of DARPA'99 tcpdump data. However, such an approach, being very simple in design, could detect a few attacks.

18. *Cannedy CMAC-based scheme*

The training time incurred in training a NN and manual updates have been the main hurdle in neural network based approaches. Therefore, Cannady proposes a method of autonomously (and automatically) learning new attacks without the need for retraining or manual updates [26]. In his proposal, a form of adaptive NN known as *cerebellar model articulation controller* (CMAC) NN has been used. CMAC NN is a localized three-layer feedforward form of NN that is designed to produced a series of input-output mappings. While the CMAC possesses excellent local generalization capabilities, it is limited in its ability to generalize universally. For this reason CMAC NNs have been applied most often to real-time robotics and signal processing. Traditional least mean squre (LMS) learning algorithm was used to update the CMAC weights (w) based on multiplying the difference in the desired output O_d and actual output O_a by a positive learning factor b as:

$$w_{i+1} = w_i + b(O_d - O_a)$$

This implementation of CMAC was designed to take feedback (s) from the system state, based on the combined effect of input packets, CPU load, available memory etc. to produce as output that represented the probability of an attack. LMS learning algorithm was modified to incorporate the feedback by using the inverse of the state of the protected system ($1 - s$) as follows:

$$w_{i+1} = w_i + (1 - s)((1 - s) - O_a)$$

For experiments, *ping flood and UDP packet storm* attack was used. During the first few iterations, the intensity of ping packets was kept slow and then gradually increased. initially, CMAC responded to the attack with 97.67% error, but after few more learning, the error got stabilized to $1.94^{-7}\%$. Similarly it could detect the similar attack called *UDP packet storm*, without training to this type of attack. This shows that CMAC is able to learn similar patterns.

19. *LNH Dynamic ID SOM scheme*

The idea of unsupervised learning is applied to NIDS by using SOM [123]. In this approach, the feature vector consists of six basic TCP features of DARPA'98 data. The training of SOM is done with about 2% of normal DARPA training data and testing is done with 10% of testing data. The attacks are detected by observing that the best matching units (BMUs) in the second layer of SOM are different in case of normal and attack data. The approach performs well by detecting 93.89% of attacks with 34.04% false positive. It is, however, not clear that why this approach is named as *dynamic approach*.
Many references to other approaches based on ANNs can be found in the [27].

2.5 Research Based on Artificial Immune System

The Human Immune System (HIS) is, perhaps, the most robust intrusion detection and prevention system, we can find in real world. Therefore many efforts have been made to imitate HIS in the area of computer security by producing Artificial Immune System (AIS).

Somayaji *et al* [177] articulate a broad vision for the development of computer immune systems by using the vertebrate immune system as a source of inspiration. The architecture of the HIS is multilayered, with defence provided at many layers. The outer most layer consists of skin and then physiological conditions like pH and temperature. Once the *pathogens* have entered the body, they are handled by *innate* and *adaptive* immune systems. Innate immune system consists of scavenger cells that clear the system from debris and pathogens. Adaptive immune system is more sensitive and involves many cells. It can be viewed as a distributed detection system which consists primarily of *white blood cells*, called *lymphocytes*. Lymphocytes can be viewed as *negative detectors*, because they detect *non-self* patterns and ignore *self* patterns, forming a bond between pathogen and receptors that cover the surface of lymphocytes. Such training is imparted in a organ called *thymus*. Only matured cells are allowed to navigate. Protection is made more specific by learning and memory. If immune system detects an unseen pathogen, it undergoes a primary response, during which it *learns* the structure of the new pathogen and updates its memory for its subsequent occurrence. Adaptive immune system also contains a molecule, called *major-histocompatibility complex* (MHC), which enables the immune system to detect intracellular pathogens(e.g. virus) that reside inside the cells. Otherwise, for lymphocyte, it is not possible to *see* inside a cell.

In the view of above paragraph, on a computer system, any corrupted data can be considered as non-self and uncorrupted data as self. If every process in a computer is viewed as cell, then a lymphocyte process can be implemented, which can query other processes to see whether they are functioning properly. Such a system can adapt to user's behavior as well. Similarly, another approach is to think each computer as corresponding to an organ and process as cell. In this model, innate immune system is composed of host-based security mechanisms, combined with network security mechanisms, like cryptographic protocols and firewalls. The adaptive immune layer can be implemented by kernel-assisted lymphocyte processes, which can navigate through computers. Each computer can work as thymus for the network by selecting lymphocytes which search for specific patterns of abnormal behavior. Despite the appealing source of inspiration, the authors in [177] address five issues that are supposed to be part of computer security system and are not directly tackled by the immune system: confidentiality, integrity, authentication, availability and accountability. They propose that the vertebrate immune system is primarily concerned with survival, viewed as a combination of integrity and availability. The following table 4 provides the mapping between immune system and corresponding computer elements for AIS.

Immune System	**Computer System**
Cell	Process (active) in a computer
Multicellular organism	Computer running multiple processes
Adaptive immunity	Lymphocyte process(s) able to query other processes to seek for abnormal behavior
Innate immune system	host-based security mechanisms combined with network security mechanisms
Self	Normal behavior of processes or files
Nonself	Abnormal behavior of processes or files
Antibody	Agent that monitors host files and neutralizes an infected file
Killer T-cell	Agent that removes the files altered by viruses and neutralized by antibody agents
Helper T-cell	Agent that controls the processes of the anti-virus system

Table 4: Mapping between immune and computer system

There has been a rapid growth in the interest and work undertaken in this field, particularly in the last five years. Two areas of computer security which have been of considerable interest to people researching AIS have been the elimination of computer viruses and worms and the detection of network attacks. An intuitive thought by many when considering AIS is "can the idea of the natural defense mechanism be mapped into a computational domain?"

20. *FHS negative selection scheme*

In [59], Forrest *et al* present a process, called "negative selection" to generate T cells as detectors. All T cells have binding regions that can detect antigen fragments (peptides). Given that the binding regions, called receptors, are created randomly, there is a high probability that some T cells detect self peptides. Such T cells are destroyed by the immune systems and remaining ones are al,lowed to get mature for future detection of antigen. Once in a circulation, if a T cell binds to antigen in sufficient concentration, a malicious event can be said to have occurred.

21. *Kephart antibody fuzzy scheme*

In an attempt to create a computer immune system for both computers and networks of computers, Kephart [92] proposes a novel immune inspired approach. The proposed AIS is capable of developing antibodies to previously unknown computer viruses or worms. These antibodies are then capable of extracting information (learning) about these new viruses and remembering them so as to recognize and respond faster in future infections. In the implementation, a particular virus is recognized via an exact fuzzy match to a relatively short sequence of bytes occurring in the virus. For this purpose, integrity monitors are used to check for any changes to programs and data files; they have a notion of self, that is any difference between the original and current version of a file is flagged.

22. *OI agent scheme*

Okamoto and Ishida propose a distributed approach to computer virus detection and neutralization by using autonomous and heterogeneous agents [150]. Their system detects viruses by matching "self" with the current host files. In this case, the self-information is a set of data characterizing the identity of the host file, such as the first few bytes of the header of a file, and the file size and path. Thus, a set of antibody agents containing the self-information was represented in a Symbolic shape-space. They are also responsible of neutralizing a detected change by overwriting the self-information on the infected files. Some types of agents, including antibody agents, are devised, inspired by the immune system.

23. *KB NIDS immune scheme*

Kim and Bentley [93] reviewed the analogy between HIS and NIDS and identified three fundamental requirements for the designing of network-based IDS: *distribution, self-organization*, and *being lightweight*. Table 5 lists the components of HIS for the requirements. In the further work [94], Kim and Bentley suggested that a generic AIS for NIDS would be

Immune System	**Characteristics of an IDS**
Immune network Unique antibody sets	Distribution
Gene library evaluation Negative selection Clonal selection	Self-Organization
Approximate recognition Memory cells Gene expression	Lightweight

Table 5: Components of the HIS that satisfy the requirement for the development of an efficient NIDS

comprised of three distinct evolutionary stages: *negative selection, clonal selection,* and *gene library evolution*. Table 6 summarizes the immune metaphors devised for their NIDS.

Immune System	**Network Environment**
Bone marrow and thymus	Primary IDS that generates the detector sets
Secondary lymph nodes	Local hosts
Antibodies	Detectors
Antigens	Network intrusions
Self	normal activites
Nonself	Abnormal activites

Table 6: Mapping between NIDS and immune system

24. *Dasgupta mobile agents scheme*

In the work by Dasgupta [36], an agent-based system for intrusion/anomaly detection is proposed. In this approach, mobile agents roamed around the nodes and routers of a network, monitoring its situation. The authors observed that the most appealing properties of the proposed agents are mobility, adaptability, and collaboration. Table 7 summarizes the immune metaphors used.

Immune System	**Network Environment**
Lymphokines	Agents that serve as message carriers or negotiators throughout the network
Helper T-cell	Agents that report the status of the environment to the user or display the decision report
Killer T-cell	Agent that takes a drastic action in case of intrusion
Suppressor T-cell	Agent that suppresses the actions of other agents
Lymphocyte trafficking	Mobile agents that navigate through the network
Co-stimulation	Signal sent by agent in the network in order to confirm an anomaly

Table 7: Mapping between NIDS and immune system

25. *GLPS layered immune scheme*

Gu *et al* [70] proposed a detection and elimination system, called the antibody layer, against Internet hackers and viruses, termed Internet antigens. Under the AIS perspective, three major parts compose the system: a database of known Internet antigens and some of their features; an evolvable antibody layer that allows the system to detect unknown antigens; and an anti-antigen procedure responsible for controlling the antibody layer and to prescribing a form of antigen elimination. Table 8 summarizes the immune metaphors used by the system.

Immune System	**Network Environment**
Host	Computer over the Internet
Antigens	Hackers, viruses
B-cells	Scanning mechanism that searches for specific antigen
Helper T-cell	A database containing information about known antigens such as suspicious behavior, that will be used to create new antigenic patterns to be presented to the B-cell
Suppressor T-cell	Host alliance that alleviates the load of the host security procedures
Anti-antigen procedure	Controls the antibody layer and provides the antibody prescription

Table 8: Mapping between antibody layer and network environment

26. *KB negative selection NIDS scheme*

It is observed by Kim and Bentley in [95] that HIS is more complex than just negative selection and they evaluate this with respect to AIS, investigating performance and scaling related to NIDS. The work builds on the LISYS system [78] and work by Kim and Bentley [96]. The profiles of TCP packet headers are used for training and testing. Thirteen *self profiles* are constructed and detectors are generated using negative selection against these profiles. The encoding of the detectors contains a number of alleles represented by an arbitrary numerical value. The different alleles on a chromosome are related to different properties of the packet. This range of values is then subjected to a clustering algorithm. The similarity between self strings and incoming strings in case of test data is measured using an *r-contiguous bit scheme*, where the value of *r* is chosen after estimating the expected number of detectors, detector generation trials and expected false negative rate. A matching activation threshold is derived from the number of detectors generated. The empirical results shows that negative selection produces poor performance due to scaling issues on real-world problems.

27. *DG negative positive selection scheme*

Dasgupta and Gonzalez, in [37], investigate and compare the performance of negative and positive selection algorithms. Positive selection is not found in the selection of T-cells in the natural immune systems, whereas negative selection is. They work from a time-series perspective in terms of scalability and changing self. Their implementation of the positive selection algorithm generates self using training data and time windows. They use a k-dimensional tree, giving a quick nearest neighbor search. At first, they use only one parameter at once, either bytes per second, packets per second or ICMP packets per second. This is then followed by a combination of all three parameters. An alert is generated when values go beyond a threshold. Their negative selection implementation uses real-valued detectors, with self defined as in their positive selection algorithm. A multi-objective genetic algorithm is used to evolve rules to cover non-self, with fitness correlated to the number of self samples covered, area and overlap with other rules. This allows for niching in the multi-objective problem. They define a variability parameter, v, as the distance from self still considered normal. This results in one rule for time windows equal to 1 and 25 rules for time window equal to 3. These rules are then used to build detectors. In their experimentations on DARPA'99 data, they consider only five attacks. Using a combination of all three parameters, all five attacks were detected. A single parameter yielded detection in 3 out of 5 cases. Positive selection needs to store all self samples in memory and is not scalable, but has very high detection rates compared to negative selection,which has a rate of 60% and 80% for window sizes of 1 and 3 respectively, using 1/100 of the memory of positive selection. Overall, the best detection rates they found were 95% and 85% for positive and negative selection respectively. They concluded that it is possible to use negative selection for IDSs, and that in their time series analysis, the choice of time window was imperative.

28. *ABCKM Danger theory based scheme*

Inspired by the previous work in immunology by Matzinger [136] and work on attack correlation by Kim and Bentley [95], in a very recent proposal by Aickelin *et al* [4], the applicability of *danger theory* to intrusion detection is discussed. They aim to build a computational model of danger theory which they consider important in order to define, explore, and find danger signals. From such models they hope to build novel algorithms and use them to build an intrusion detection system with a low false positive rate. The correlation of signals to alerts, and also of alerts to scenarios, is considered particularly important. Their proposed system collects signals from hosts, the network and elsewhere, and correlates these signals with alerts. Alerts are classified as good or bad in parallel to biological cell death by apoptosis and necrosis. Apoptosis is the process by which cells die as a natural course of events, as opposed to necrosis, where cells die pathologically. It is hoped that alerts can also be correlated to attack scenarios. Where these signals originate from is not yet clear, and they will probably be from a mixture of host and network sources. Examples could include traffic normalizers, i.e. a device which sits in the traffic stream and corrects potential ambiguities in this stream, packet sniffers etc. The danger algorithm is, however, yet to be specified, as is the correlation algorithm. Whether the system will actively respond to attacks is also not yet clear. Aickelin *et al* conclude that if this approach works, it should overcome the scaling problems of negative selection, but that a large amount of research still remains to be done. In the future, they intend to implement such a system.

2.6 *Research Based on Data Mining Approaches*

The problem of IDS can also be considered as problem of pattern recognition and classification. From this standpoint, various data mining techniques have been applied to build efficient IDSs. The following categorization lists some of the techniques and the purposes for which they may be employed.

- Characterisation/Generalisation - Produces a general description of the nature of the data. It can be used for deviation analysis if adequate difference is present in data content.
- Classification - Creates a categorization of data records. It could be used to detect individual attacks, but as described by previous experiments in the literature, it is prone to produce a high false alarm rate. This problem may be alleviated by applying fine-tuning techniques such as boosting .
- Association - Describes relationships within data records. Detection of irregularities may occur when many records exhibit previously unseen relationships.
- Frequent Episodes - Describes relationships in the data stream by recognizing records that occur together. For example, an attack may produce a very typical sequence of records (similar to system call traces used in the AIS-based approaches). This technique may produce results for distributed attacks or attacks with arbitrary noise inserted within.
- Clustering - Groups records that exhibit similar characteristics according to some pre-defined metrics. It can be used for general analysis similar to classification, or for detecting outliers that may or may not represent attacks.

- Incremental updates - Keeps rule sets up-to-date to better reflect the current regularities in the data.
- Meta-rules - Provides a description of changes within rule sets over time. It can be used for trend analysis. It is probably not very useful for detecting individual attacks, but can serve as a base of comparison when new, unusual rules appear. For example, if a trend shows a steady increase in some network activity over time, then a sudden increase in another type of activity may be suspicious, but not necessarily an increase in the former.

This section overviews various data mining techniques applied to IDS, irrespective of data sources (i.e. network-based or host based).

2.6.1 Association Rules and Clustering Based Approaches

This section overviews some of the major work in intrusion detection using rule-based and cluster-based approaches.

29. *MADAM ID*

At University of Columbia, the work of Lee *et al* [110][111][112][113][114] [115][116][117] makes the extensive use of various data mining techniques to build efficient intrusion detection model. The central theme of their approach is to apply data mining methods to the extensively gathered audit data to compute models that accurately capture the actual behaviour (patterns) of intrusions and normal activities. This automatic approach eliminates the need to manually analyze and encode intrusion patterns, as well as the guesswork in selecting statistical measures for normal usage profiles. The resultant framework, called MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) consists of programs for learning classifiers and meta-classification, association rules for link analysis, frequent episodes for sequence analysis, and a support environment that enables system builders to interactively and iteratively drive the process of constructing and evaluating detection models. For classification task, RIPPER algorithm [32] is used to induce rules which work as patterns for activities. The goal of association rules is to derive multi-feature correlations from a database table. To avoid the generation of uninterested rules, the concept of *axis* and *reference* attributes is proposed. Some attributes are essential in describing the data, while others only provide auxiliary information. These essential attributes are termed as *axis attributes*. Similarly, another interesting characteristic of system audit data is that some attributes can be the references of other attributes. These reference attributes normally carry information about some "subject", and other attributes describe the "actions" that refer to the same "subject". Thus, an association rule is valid if it contains at least one *axis* attribute. Frequent episodes are used to represent sequential audit record patterns. Various modifications are made to original frequent episode algorithm. The extended algorithm computes frequent sequential patterns in two phases:

- It finds the frequent associations using axis features.
- It generates the frequent serial patterns from these associations.

Thus the approach combines the associations among features and the sequential patterns among records into a single rule.

. It is often necessary to include the low frequency patterns i.e network activities, which do not occur frequently, but do occur in the network (e.g. finger or gopher). We need to include their patterns into the network traffic profile. If we use a very low *support value* association mining algorithms, we will then get unnecessarily a very large number of patterns related to the high frequency services, for example, smtp. In order to overcome this difficulty, a level-wise approximate mining procedure for finding frequent sequential patterns from audit data is proposed, which is show in figure 3. Often, the *interestingness* of the patterns (rules, so obtained) is measured by using the minimum support and confidence values to output only the *statistically significant* patterns. The basic algorithms implicitly measure the *interestingness* (i.e., relevancy) of patterns by their support and confidence values, without regard to any available prior domain knowledge. That is, if I is the interestingness measure of a pattern p, then

$$I(p) = f(support(p), confidence(p))$$

```
Input: the terminating minimum support $s_0$;
the initial minimum support $s_i$;
the axis attribute(s);
Output: frequent episode rules Rules
Begin
    $R_{restricted} = \phi$
    scan database to form $L = \{1 - itemsetsthatmeets_0\}$
    $s = s_i$;
    while ($s \geq s_0$) do begin
        compute frequent episodes from $L$:
        each episode must contain at least one axis attribute value that is not in $R_{restricted}$;
        append new axis attribute value to $R_{restricted}$;
        append episode rules to the output rule set $Rules$;
        $s = \frac{s}{2}$ /* smaller support value for the next iteration */
    end while
end
```

Figure 3: Level-wise Approximate Mining Procedure

where f is some ranking function. The above relation is modified to incorporate schema level information into the interestingness measures, as follows:

$$I_e(p) = f_e(I_A(p), f(support(p), confidence(p))) = f_e(I_A(p), I(p))$$

where I_A is a measure on whether a pattern p contains the specified important (i.e. "interesting") attributes and f_e is a ranking function that first considers the attributes in the pattern, then the support and confidence values. Therefore, for the axis attributes, we have

$$I_A(p) = \begin{cases} 1 & \text{if p contains axis attributes;} \\ 0 & \text{otherwise.} \end{cases}$$

One of the main contributions of this study is the construction of feature vectors for network level data. Each connection is described in terms of *intrinsic features* (like source IP and port number and destination IP and port number etc) and *processed features* like *same_host*, *same_service* etc within a defined time interval. This second category is very useful in detecting various DoS attacks. The KKD'99 data set is derived by using this approach. The proposed approach is evaluated on DARPA'98 data and in all, it could attain a detection rate of 68% with false positive rate of 2%. The same approach is also use to detect the problem of masquerading. Under this approach, frequent patterns from user command data are mined and merged to into an aggregated set to form normal usage profile of a user. Any new pattern set is compared with the profile and a similarity, $\frac{m}{n}$, score is assigned, where n is total patterns in the set and m patterns have the match. Two pattern are said to have matched if they have same left-hand-sides and right-hand-sides, their support values are within a 5% of each other and their confidence values are also within 5% of each other. Using the proposed frequent episode algorithm, the approach could differentiate among various user with high accuracy.

30. *ADAM*

On the similar lines, Barbará *et al* propose a testbed called ADAM (Audit Data Analysis and Mining) for using data mining techniques to detect intrusions [11][12]. Under training, the system is trained on network data using only basic TCP features. ADAM uses connections as the basic granule, obtaining the connections from the raw packet data of the audit trail. This preprocessing results in a table with the following schema:
$R(T_s, Src.IP, Src.Port, Dst.IP, Dst.Port, FLAG)$,
where T_s represents the beginning time of a connection, $Src.IP$ and $Src.Port$ refer to source IP and port number respectively, while $Dst.IP$ and $Dst.Port$, represent the destination IP and port number. The attribute $FLAG$ describes the status of a TCP connection. The relation R contains the dataset on which association mining is performed. The itemsets that are *aggregations* of source IP or Port values, e.g., connections that come from a source domain and have the same destination IP, are also considered and termed as *domain-level* itemsets. (For instance, we want to discover frequent connections from Source IP X to Destination IP Y, or from Source Domain W to Destination IP Y.)
First, ADAM is trained using a data set in which the attacks and the attack-free records are labeled. The following steps are involved.

- A database of frequent itemsets (those that have support above a certain threshold) for the attack-free portions of the data set is created. This serves as a profile against which frequent itemsets found later will be compared.

- The profile database is populated with frequent itemsets as the aforementioned schema, as well as frequent *domain-level* itemsets for attack-free portions of the data.
- The itemsets in this profile database are cataloged according to the time of the day and day of the week, to further refine the specificity of these rules to variations of workload during the different time periods.

The mining algorithm used for this first step of the training phase is an *off-line* algorithm. To complete the training phase, an incremental, *on-line* algorithm to detect itemsets that receive strong support within a period of time is used. This algorithm is driven by a sliding window of tunable size k. The algorithm outputs itemsets that have received strong support during this window. An itemset that starts receiving support, is compared with itemsets in the profile database for an analogous time and day of the week. If the itemset is present in the profile database, leave it at this stage and take another itemset. On the other hand, a support counter is kept if the itemset is not in the database. If the itemset's support surpasses a threshold, that itemset is reported as suspicious. For a set of suspicious itemset, two services are provided . First the ability to drill down and find the raw data in the audit trail that gives rise to these rules. Secondly, suspicious itemsets is annotated with a vector of parameters (based on the raw audit trail data that gave rise to the rules). Since we know where the attacks are in the training set, the corresponding suspicious itemsets along with their feature vectors are used to train a classifier. The trained classifier will be able to, given a suspicious itemset and a vector of features, classify it as a known attack (and label it with the name of the attack), as an unknown attack (whose name is not known), or as a false alarm. ADAM is then ready to detect intrusions online. Again, the on-line association rules mining algorithm is used to process a window of the current connections. Suspicious connections are flagged and sent along with their feature vectors to the trained classifier, where they are labeled accordingly. The approach is tested on DARPA data for DoS and Probing attacks categories and attains a detection rate between 30% and 47% with 100 false alarms per day for prob and DoS attacks. It should be noted, however, that the approach is same as that of Lee *et al*, except the use of rule learner RIPPER in Lee scheme.

31. *ADMIT*

The problem of masquerading is also addressed by Sequeira and Zaki by forming the user profiles by means of clustering [173]. They propose a host-based IDS called ADMIT (Anomaly Data Mining for InTrusion). ADMIT is a user-profile dependent, temporal sequence clustering based, real-time intrusion detection system with host-based data collection and processing. There are two stages: Training phase in which user profiles are created and testing phase in which user's command stream is verified against the user's profile. A user's command history is recorded and converted into the tokens(i.e. individual command in parsed form). A sequence is a set of l tokens. These sequences are grouped into clusters by using a clustering algorithm, called *DynamicClusting*. ADMIT has following components. *ProfileManager* is responsible for security of a terminal. It has three components: *ProfileCreator* creates profile for users authorized to use the terminal and *ProfileUpdater* updates profile for those users, while *SequenceExaminer* examines each sequence of tokens of process data and determines if that is characteristic or not, of the user thought to have produced it. *ProfileCreator* and *ProfileUpdater* use two submodules. *FeatureSelector* parses the data into tokens and then converts tokens into sequence of tokens. *ClusterCreator* converts the input array of sequences of tokens into clusters which form the profile of the user they originate from. Clusters can be represented by the cluster center, which is defined as the sequence having the maximum aggregated similarity to the other sequences within the cluster, i.e. if $c = \{s_0, s_1, \ldots, s_{n-1}\}$ is a cluster with n sequences, then

$$s_c = max_{s_i \in c}\{\sum_{j=0}^{n-1} Sim(s_i, s_j)\}$$

Therefore a user's profile is a set of clusters. Thus for a user u, its profile is:

$$p_u = \{c_i | (Sim(s_c, s) \geq r, \forall s \in c_i) \wedge (Sim(s_{c_i}, s_{c_j}) \leq r', i \neq j)\}$$

where r and r' are untra-cluster and inter-cluster similarity thresholds, respectively, s_c is the center of cluster c_i, and $Sim(s_1, s_2)$ is the similarity between two sequences. The pseudo-code of *DynamicClusting* is given in figure 4. These clusters are further refined by

```
**DynamicClustering**($r, S_u, p_u, S_u^c$):
//$r$ is intra-cluster similarity threshold
//$s_u$ is a set of a user u's sequences to be clustered
//$p_u$ is the user u's profile
//$S_u^c$ is the set of user u's cluster centers
$S_u^a = S_u$ //set of user u's anomalous sequences
while $(S_u^a \neq \phi)$
    select random $s_c \in S_u - S_u^c$ as new cluster center
    $c_{new} = \{s_c\}$ // initialize new cluster
    $S_u^c = S_u^c \cup s_c$
    for all remaining sequences $s_i \in S_u - S_u^c$
        if $(Sim(s_i, s_c) \geq r)$
            if $(Sim(s_i, s'_c) < Sim(s_i, s_c) \forall s'_c \in S_u^c - s_c)$
                $c_{new} = c_{new} \cup \{s_i\}$
                recalculate cluster center, $s_c$ for $c_{new}$
    $p_u = p_u \cup c_{new}$
    $S_u^a = S_u^a - c_{new}$
```

Figure 4: Pseudo code of the *DynamicClustering* algorithm

merging or splitting them for better accuracy. The pseudo-codes for algorithms are given in figure 5.

In the testing phase, the incoming sequence from a user is matched against its profile by calculating the similarity. These sequences are, then, rated using different sequence rating schemes. For example, the rating R_j for the jth sequence is calculated as

$$R_j = \alpha * Sim(s'_j, p_u) + (1 - \alpha) * R_{j-1}$$

where $R_0 = Sim(s'_0, p_u)$, $0 \leq \alpha \leq 1$. A threshold value, T_{ACCEPT}, is used on the rating of a sequence to determine if it is normal or not. If a sequence is below the threshold, it is said to have come from a masquerader, otherwise from a genuine user. In order to accommodate the behavior change, the idea of incremental clustering is proposed. If the size of a cluster increases beyond a certain threshold, $T_{cluster}$, a different type of alarm, called B alarm, is raised. The *IncrementalClustering* algorithm is shown in figure 6. It has been observed that smaller sequence length (5) gives more accurate results. Also a data size of 125 session is found to be enough to learn the profile. The concept of real-time learning is also tested by merging the sequences into the existing profiles by creating clusters. This

```
**MergeClusters**$(r', p_u, S^c_u)$:
//$r'$ is inter-cluster similarity threshold
for each pair of clusters $c_i, c_j$ in profile $p_u, i \neq j$
    if $(Sim(c_i, c_j) \geq r')$
        $c_i = c_i \cup c_j$ //merge cluster
        recalculate center for $c_i$
        $p_u = p_u - c_j$
        $S^c_u = S^c_u - s_{c_j}$
**SplitClusters**$(r, t_s, p_u, S^c_u)$:
//$t_s$ is a spliting threshold support
for each cluster $c_i$ in profile $p_u$
    if $(cluster\_support(c_i) > t_s)$
        DynamicClustering$(r + 1, c_i, p_u, S^c_u)$
        $p_u = p_u - c_i$
        $S^c_u = S^c_u - s_{c_i}$
```

Figure 5: Pseudo codes of the *MergeCluster* and *SlpitCluster* algorithms

could decrease the false positive, but results in high false negative. In all, the approach achieves 80% detection rate with 15% false positive rate.

32. *MINDS*

A team of researchers at university of Minnesota investigates the usability of various data mining and machine learning approaches under the project *The MINDS- Minnesota Intrusion Detection System* [52, 48, 107, 108]. The research concentrates on the following three integral parts:

- Anomaly detection algorithms
- Summarization of attacks using association pattern analysis
- Learning from rare class-building rare class prediction models

The proposed approach builds model on network data. For real time detection, the data is collected from Cisco routers using Netflow utility. Feature vectors are constructed using two types of feature- (a) Time-window based features and (b) Connection-window based features. The choice of feature construction is same as that used by Lee *et al* [113]. Under the MINDS *anomaly detection module*, the following schemes are used (evaluated):

```
**IncrementalClustering**($s'_i, S''_a, r, r_i, r'_i, p_u, S^c_u$):
//$s'_i$ is an anomalous sequence
//$S''_a$ is the list of anomalous sequences
//$r_i$ and $r'_i$ are intra and inter incremental cluster proximity threshold
//$r$ is intra-cluster similarity threshold
//$p_u$ is the user u's profile, being updated incrementally
//$S^c_u$ is the set of user u's cluster centers
if the maximum difference in adjacent sequence ids of $S''_a < r'_i$ and the mean difference in list of sequence ids $< r_i$
    $S''_a = S''_a \cup s'_i$
    if $(|S''_a| > T_{cluster}$
        Raise an alarm of type B
else
    $DynamicClustering(r, S''_a, p_u, S^c_u)$
    $S''_a = \{s'_i\}$
```

Figure 6: Pseudo code of the *IncrementalClustering* algorithm

LOF-Local Outlier Factor algorithm The idea is to assign to each data point a degree of being outlier, called *local outlier factor*. The outlier factor is local in the sense that only a restricted neighborhood of each object is considered. The LOF is originally defined in [22].

Nearest Neighbor (NN) approach The idea is to calculate the distance of each data point from its k-th NN and if this distance is greater than a threshold, the point under consideration is an outlier. Under this study, $k = 1$.

Mahalanobis-distance based approach The Mahalanobis distance between a point p and the mean μ of the normal data is calculated as:

$$d_m = \sqrt{(p-\mu)^T \sum^{-1} (p-\mu)}$$

where $\sum$ is the covariance matrix of the normal data. The point p is considered as outlier if this distance is greater than a threshold.

Unsupervised SVMs The unsupervised SVMs, proposed in [164], are used to separate the unlabelled data into regions. The regions with low density are considered as outliers.

In case of known anomalies (i.e. misuse-based approach), it can be noted that anomalies are from minority class, when compared with whole normal data. Therefore, learning algorithms should be well suited for this skewed distribution. SMOTEBoost is such an algorithm, used in this work, to learn from rare class i.e. known anomalies [30]. SMOTEBoost algorithm combines the Synthetic Minority Oversampling Technique (SMOTE) [29] and the standard boosting procedure [62]. *ROC analysis* and metrics such as *precision*, *recall* and *F-value* have been used to understand the performance of the learning algorithm on the minority class.

$$\begin{aligned} Precision &= \frac{TP}{(TP+FP)} \\ Recall &= \frac{TP}{(TP+FN)} \\ F-value &= \frac{(1+\beta^2)\cdot Recall\cdot Precision}{\beta^2\cdot Recall+Precision} \end{aligned}$$

where β corresponds to relative importance of *precision* vs. *recall* and it is usually set to 1. The main focus of all learning algorithms is to improve the *recall*, without sacrificing the *precision*. However, the recall and precision goals are often conflicting and attacking them simultaneously may not work well, especially when one class is *rare*. The *F-value* incorporates both precision and recall, and the "goodness" of a learning algorithm for the minority class can be measured by the *F-value*. While ROC curves represent the trade-off between values of TP and FP, the F-value basically incorporates the relative effects/costs of recall and precision into a single number.

STOME algorithm works as follows:

- For the continuous features
  - Take the difference between a feature vector (minority class sample) and one of its k nearest neighbors (minority class samples).
  - Multiply this difference by a random number between 0 and 1.
  - Add this difference to the feature value of the original feature vector, thus creating a new feature vector
- For the nominal features
  - Take majority vote between the feature vector under consideration and its k

nearest neighbors for the nominal feature value. In the case of a tie, choose at random.

– Assign that value to the new synthetic minority class sample.

Using this technique, a new minority class sample is created in the neighborhood of the minority class sample under consideration. The neighbors are proportionately utilized depending upon the amount of SMOTE. Hence, using SMOTE, more general regions are learned for the minority class, allowing the classifiers to better predict unseen examples belonging to the minority class.

The SMOTEBoost algorithm, shown in figure 7, proceeds in a series of T rounds. In every round a weak learning algorithm is called and presented with a different distribution D_t altered by emphasizing particular training examples. The distribution is updated to give wrong classifications higher weights than correct classifications. Unlike standard boosting, where the distribution D_t is updated uniformly for examples from both the majority and minority classes, in the SMOTEBoost technique the distribution D_t is updated such that the examples from the minority class are oversampled by creating synthetic minority class examples. The entire weighted training set is given to the weak learner to compute the weak hypothesis h_t. At the end, the different hypotheses are combined into a final hypothesis h_{fn}. These clusters are further refined by merging or splitting them for better accuracy. The pseudo-codes for algorithms are given in figure 5. SMOTEBoost algorithm generates the artificial examples from the minority (intrusion) class within the boosting step. Artificial examples are created after each boosting round, classifiers are then built on such newly generated data and finally they are combined using the boosting technique. The classifier represents the rules as association-rules. The association rules are formed using the degree of discrimination of a pattern. In other words, let a set of features X that occurs c_1 times in anomalous class and c_2 times in normal class, n_1 be the number of anomalous connections in the data set, n_2 be the number of normal connections in the data set, then the ratio $\frac{c_1/n_1}{c_2/n_2}$ would indicate how well the pattern X could discern anomalous connection from normal connections. Similar to KDD'99 data set, a network level data set is derived from

```
Given: Set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$,
where $C_m$, ($C_m < C$) corresponds to a minority class.
Define $B = \{(i, y) : i = 1, \ldots, m,\ y \neq y_i\}$
    Initialize the distribution $D_1$ over the examples, such that $D_1(i) = 1/m$.
for $t = 1, 2, \ldots, T$
    - Modify distribution $D_t$ by creating $N$ synthetic examples from minority class $C_m$ using the
    SMOTE algorithm
    - Train a weak learner using distribution $D_t$
    - Compute weak hypothesis $h_t : X \times Y \to [0, 1]$
    - Compute the pseudo-loss of hypothesis $h_t$:
      $\varepsilon = \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$
    - Set $\beta_t = \varepsilon_t/(1 - \varepsilon_t)$ and $w_t = 1/2(1 - h_t(x_i, y) + h_t(x_i, y_i))$
    - Update $D_t$ :  $D_{t\backslash 1}(i, y) = (D_t(i, y)/Z_t)\beta_t^{w_t}$
      where $Z_t$ is a normalization constant chosen such that $D_{t\backslash 1}$ is a distribution.
Output the final hypothesis: $h_{fn} = arg\ max_{y \in Y} \sum_{t=1}^{T} (log \frac{1}{\beta_t}) h_t(x, y)$
```

Figure 7: The SMOTEBoost algorithm.

DARPA'98 data to evaluate the proposed framework. Under the anomaly-based architecture, on an average, LOF based approach outperforms the remaining three approaches. On the real network data, the proposed work is able to identify many zero-day attacks.

2.6.2 Statistical Approaches

Statistical approaches are extensively used in data mining task, especially for unsupervised learning. Denning was among the first to propose a detailed model for anomaly-based system, based on statistical techniques [42]. In this approach, profiles of subjects (users) are learnt by monitoring the system's audit records and statistical methods (means and standard deviation) used to calculate deviations from the normal behavior. Any significant deviation in the normal behavior is consider as a possible intrusive event.

33. *PHAD, ALAD, LERAD*

Based on conditional probability, Mahoney and Chan propose PHAD (packet header anomaly detection), ALAD (application layer anomaly detection) and LERAD (learning rules for anomaly detection) systems for network-based analysis [130][131][132]. PHAD is based on the assumption that events that occur with probability p should receive an anomaly score of $\frac{1}{p}$. PHAD uses the rate of anomalies during training to estimate the probability of an anomaly in detection mode. If a packet field is observed n times with r distinct values, there must have been r anomalies during the training period. The probability that the next observation will be anomalous is approximated by $\frac{r}{n}$. In addition to this, PHAD weights the probability of occurrence of an event by the last time it occurred by using *non-stationary model* i.e. if an event last occurred t seconds ago, then the probability that it will occur in the next one second is approximated by $\frac{1}{t}$. Each packet header field containing an anomalous value is assigned a score inversely proportional to the probability i.e.

$$score_{field} = \frac{tn}{r}$$

Finally, we get the anomalous score of entire packet as:

$$score_{packet} = \sum_{i \in anomalous\_field} \frac{t_i n_i}{r_i}$$

where anomalous fields are the packet fields with values not found in the training model. Since storing all the different observed values is very expensive, an approach based on storing ranges instead of individual values is proposed. A maximum limit, C, for list of ranges is fixed, in such a way that if C is exceeded, then the method finds two closest ranges and merges them. On DARPA'99 data, the proposed method attains a detection ratio of 72 over 201 attacks with a rate of 10 alarms per day ($C = 32$).

ALAD is a system that introduces the conditional rules to TCP connections. After testing 15 different combinations of features, the authors find the following 5 forms of models appropriate:

I. Pr(src IP|dest IP), which learns the normal set of clients for each host.

II. Pr(src IP|dest IP, dest port), which learns the normal set of clients for each server, which may be different across the servers on a single host.

III. Pr(dest IP, dest Port), which learns the set of local servers which normally receive requests.

IV. Pr(TCP flags|dest Port), which learns the set of normal TCP flag sequences.

V. Pr(keywords|dest Port), which learns the normal set of keywords in requests to different services.

These probabilities are estimated during the training period.

LERAD improves the performance by combining two aforementioned approaches by using a rule-learning algorithm as follows:

- Generate rules with initial $\frac{n}{r} = \frac{2}{1}$ on L randomly selected pairs of training examples.
- *Coverage test*: remove rules that do not predict any value(on a small sample) not already predicted by rules with higher $\frac{n}{r}$.
- Train on the full training set, removing any rule that generates anomalies in the last E examples.

On DARPA'99 dataset, LERAD detects an average of 114 out of 190 attacks, or 60%, at 100 false alarms (10 per day).

34. *SPICE*

In [179], Staniford *et al* outline an approach to detecting network scans by examining a packet's probability. Probability is based upon traffic characteristics including the expectation of seeing the features like source IP address, source port, destination IP address and destination port. The advantage of this approach is that it creates very little processing overhead. Staniford *et al* propose an advanced method of information decay that is a function of packet normality. Thus, the more anomalous a packet, the longer it will "stay around". This technique is aimed at detecting scans/probs. Staniford *et al* present this overall system as SPICE (Stealthy Probing and Intrusion Correlation Engine). The empirical results show SPICE as being very effective at detecting both scans that make no attempt to evaded detection, and scans that do. While having a very narrow focus SPICE is quite useful as it not only detects port-scans but any type of scan. Scan detection is important in ID as a large number of network attacks include a scan component, for example, automated attack tools that scan large numbers of hosts for specific vulnerabilities and Internet worms. The SPICE system is also the only known method from the reviewed papers to be implemented in an actual IDS product– SPADE, which is a plug-in built for the freely available open source Snort IDS. A feature that is unique to SPICE which makes it particularly well suited to production environments is that it requires no training period to learn what normal traffic looks like. SPICE also has a mode in which the normality threshold is dynamically readjusted to maintain a constant level of false positives. The major failing of SPICE is that it does not examine packet payload and thus is unable to detect the whole range of intrusions that use malformed or specially crafted payloads that are otherwise part of a normal packet.

2.7 *Approaches Based on Analysis of Traffic Flow*

Traffic flows are essentially temporal series that exhibit regular patterns. Therefore methods based on stochastic processes and signal analysis have been applied in order to characterize them, and therefore, to detect anomalous behavior. These methods are efficient in identifying attacks that generate anomalies in the patterns of normal flows e.g. unusual number of packets or bytes.

35. *CRM statistical traffic modelling*

Cabrera *et al* [23] examine the role of *statistical traffic modelling* to detect novel attacks by evaluating the discriminating power of traffic measurements, rather than designing classifiers to categorize the traffic. The authors propose two models– (a) *network activity model*, to detect the presence of DoS and Porbing attacks by capturing the volume of traffic, and (b) *application model*, to detect the presence of attacks that utilize a given application e.g. `telnet` by capturing operations of such applications. The models are built on the following processes:

- $X_N^T(k)$:Total number of normal connections which were initiated in the interval $[kT, (k+1)T]$.

- $X_A^T(k)$:Total number of attack connections which were initiated in the interval $[kT, (k+1)T]$.
- $X_C^T(k)$:Total number of connections which were initiated in the interval $[kT, (k+1)T]$

These connection counting processes are the basis of *attack and normal intervals*, define as follows:

$$X_A^T(k) > X_N^T(k) \;\Rightarrow\; [kT, (k+1)T] \; attack\ interval$$
$$X_A^T(k) \leq X_N^T(k) \;\Rightarrow\; [kT, (k+1)T] \; normal\ interval$$

On the basis of experiments, it can be inferred that the normal connections $X_N^T(k)$ follow three operating regimes: a day regime, an evening regime, and a night regime. Moreover, the day and night regimes seems to adjust to Poisson processes; and attack connections $X_A^T(k)$ present heterogeneous behavior. The DoS and probing attacks appear in bursts, while other forms of attacks appear on a single connection. Several models are studied by estimating the probability density functions for each regime and setting thresholds as basic detection mechanism. The well known Kolmogorov-Smirnov test is used to achieve this purpose. The results obtained are between 60% to 80% of correct detections for day regime with no false positive.

Wavelet based network traffic data analysis has been drawing a lot of attention of many researchers. A detailed bibliography regarding the work in this direction can be found in [193].

36. *Gilbert multiscale analysis*

The study by Gilbert [66] discusses the theoretical and implementation issues of wavelet based scaling analysis for network traffic. The network traffic is characterized by packets per second and user-requested-page per session for the demonstration of the presence of self-similarity in the traffic. A process or function $\{f(t) : t \in (-\infty, \infty)\}$ is said to be self-similar with self-similarity parameter H, if and only if $\{c^{-H} f(ct) : t \in (-\infty, \infty)\}$ and $\{f(t) : t \in (-\infty, \infty)\}$, $\forall \quad c > 0$ and $t \in (-\infty, \infty)$ have same distributions or in simple terms, A process or function $\{f(t) : t \in (-\infty, \infty)\}$ is said to be self-similar with self-similarity parameter H, if and only if $c^{-H} f(ct) = f(t)$, $\forall \quad c > 0$ and $t \in (-\infty, \infty)$. The parameter H is called Hurst parameter. For a general self-similar process, the parameter H measures the degree of self-similarity. For random processes suitable for modeling network traffic, H is basically a measure of the speed of decay of the tail of the autocorrelation function. If H lies between 0.5 and 1, the process is LRD and if it lies between 0 and 0.5, the corresponding process is said to have Short Range Dependence (SRD). Hence, in the study of anomaly behaviour of traffic data, any deviation of H from 0.5 to 1 range signifies the presence of anomaly in the data.

A function $f \in L^2$ has the following wavelet representation

$$f = \sum_{k \in Z} c_{J,k}\phi_{J,k} + \sum_{j \geq J;k} d_{j,k}\psi_{j,k} = \sum_{j,k \in Z} d_{j,k}\psi_{j,k}. \tag{3}$$

Energy-plots and partition functions are calculated using the wavelet coefficients. The coefficients in the wavelet representation of a self-similar function satisfy the following simple relation: for any integers j, m, n such that $j = m + n$, we have

$$\begin{aligned} d_{j,k} &= 2^{\frac{j}{2}} \int_{-\infty}^{\infty} f(t)\psi(2^j t - k)dt \\ &= 2^{\frac{j}{2}} \int_{-\infty}^{\infty} f(2^{-n}t)\psi(2^m t - k)2^{-n}dt \\ &= 2^{\frac{j}{2}} 2^{-nH-n} \int_{-\infty}^{\infty} f(t)\psi(2^m t - k)dt \\ &= 2^{\frac{-n(2H+1)}{2}} d_{m,k}. \end{aligned} \tag{4}$$

Taking $m = 0$ and computing the energy E_j, at j^{th} scale, of wavelet coefficients, we get

$$\begin{aligned} E_j &:= \tfrac{1}{N_j} \sum_k |d_{j,k}|^2 \\ &= \tfrac{2^{-j(2H+1)}}{N_j} \sum_k |d_{0,k}|^2 \quad = 2^{-j(2H+1)} E_0 \end{aligned} \tag{5}$$

In the above equation, N_j represents the number of wavelet coefficients at scale j. From the equation (4), it may be noted that N_j is same at all levels. Consequently, the Hurst parameter H can be computed using

$$H = \frac{1}{2}\left[\frac{1}{j} log_2 \left(\frac{E_0}{E_j}\right) - 1\right] \tag{6}$$

The plots of logarithm of energies at various levels have been considered. The scales, over which the plots are straight lines, are determined to identify the scale interval over which the self-similarity holds. The presence of intrusion into the data is found by analyzing the deviation from the line behavior of energy plots at the determined scales. The presence of self-similarity in data is inferred from the straight line behavior of energy plots. The wavelet based partition function is defined as

$$S(q, j) = \sum_k |C_j d_{j,k}|^q \tag{7}$$

where C_j is a normalizing constant to remove the L^2 normalization of the wavelets. The partition function computes the qth order moments of wavelet coefficients as function of scale. A graph between log$S(q, j)$ and j is plotted which gives a family of curve, indexed by q. Intuitively, partition function collects the statistics about the local behavior of the function.

37. *NR scheme for simulation of self-similarity of network*

In another work, Nash and Ragsdale [147] propose the use of self-similarity to generate network traffic for IDS evaluation. They observe that it is difficult both to produce traffic that includes a large number of intrusions and analyze such a huge traffic for signs of intrusions. They use self-similarity to reproduce real traffic, and the wavelet coefficients to decompose the data for analysis. Hurst parameter is estimated by using Mandelbrot's method, termed as *rescaled range statistic*, abbreviated as R/S [134]. Letting $X^*(t) = \sum_{k=1}^{t} X_t$, the R/S statistic $R/S(s, t)$ for some lag s and start time t is given by

$$R/S(s, t) = \frac{R(s, t)}{S(s, t)} \tag{8}$$

where

$$\begin{aligned} \gamma(s, t, u) &= X^*(t + u) - X^*(t) - (\frac{u}{s})[X^*(t + s) - X^*(t)] && (9)\\ R(s, t) &= \max_{0 \le u \le s} \{\gamma(s, t, u)\} - \min_{0 \le u \le s} \{\gamma(s, t, u)\} && (10)\\ S(s, t) &= \sqrt{(1/s) \sum_{k=t+1}^{t+s} X_k^2 - [(1/s) \sum_{k=t+1}^{t+s} X_k]^2} && (11) \end{aligned}$$

Mandelbrots method is a graphical technique, whereby successive calculations of R/S are plotted on a $log - log$ scale for various values of lag s and start time t. The slope of a straight-line fit for these values of R/S forms an estimate of the Hurst parameter and therefore, the demonstration of self-similarity of the network data. The wavelet coefficients of the original data are calculated and according to some threshold, large coefficients are selected. The data is reconstructed and the Hurst parameter is again estimated. On DARPA data, it is shown that the Hurst parameter from the reconstructed data also confirms the presence of self-similarity. Therefore, the wavelets coefficients, which are small in number as compared to whole data, can be used to analyze the network in a simulated environment.

38. *HFW wavelets network performance problem*

Along similar lines, Huang *et al* [80] propose the use of energy plot to analyze the network (FDDI ring ISP network and Internet traffic from a research lab) in terms of RTT (round trip time) and RTO (retransmission timeout). A tool named as WIND has been built to analyze the packets collected from the tcpdump tool. TCP/IP packets can be analyzed across different time periods and across part of traffic destined to different subnets by exploiting the built-in scale-localization ability of wavelets.

39. *BKP IMAPIT*

In [13], the use of wavelet coefficients is proposed to analyse various network related anomalies. These anomalies are grouped into three categories: *Network operation anomalies* which include network device outages and change in traffic due to configurational changes; *flash crowd anomalies* which include traffic due to some software release or external interest in some specific web site and *network abuse anomalies* which include DoS or scans. The work of Barford *et al* considers the time-frequency characteristics of IP flow and SNMP data collected at the border router of the university. To automate this work, a framework called Integrated Measurement Analysis Platform for Internet Traffic (IMAPIT) is proposed [14]. IMAPIT contains a data management system which support and integrates IP flow, SNMP and anomaly identification data. IMAPIT also includes a robust signal analysis utility which enables the network traffic data to be decomposed into its frequency components using a number of wavelet and framelet systems. The key to the detection of anomalies by using this wavelet approach lies in its inherent time-frequency analysis. This property allows dividing the signal into different components at several frequencies. In the case of traffic flow, low frequency part corresponds to patten of very long duration. Mid frequency part captures daily variation in the data, and high frequency part consists of short term variation. In order to obtain these three components, wavelet coefficients are grouped onto three intervals and signals are subsequently synthesized from them. Once obtained the long-term, mid-term and short-term patterns of behavior, an algorithm termed *deviation score* is proposed to automatically identify regularities in data. By the very nature of short term anomalies, such as several forms of DoS attacks and port scans, they are detected within mid-band and high-band components. By separating these parts from long-term behavior, the decomposition facilitates to easily visualize these anomalies as isolated peaks in the upper bands. Subsequently, detection by setting thresholds can be performed.

40. *LoSS scheme*

The authors of [7] use the loss of self-similarity (LoSS) technique to characterize many DoS attacks whose effectiveness depends on a continuing stream of attack traffic (termed as DoS-TE). In [7], self-similarity is determined by calculating the Hurst parameter H, using Whittle and periodogram approaches [16], and the following conditions are used to decide the loss of self-similarity:

$$MIN(periodogram,\ Whittle) \leq 0.05$$
$$OR$$
$$MAX(periodogram,\ Whittle) \geq 0.99$$

A time series of packet's arrival count per unit time is constructed to calculate the periodogram and Whittle estimates of the Hurst parameter. A sliding window, ranging from 10 minutes to 30 minutes, is used to construct the time series. Data sets from other sources are used as normal background traffic and DARPA data is used for attacks. In order to capture burstiness, peak packets per second (pps) rate of each attack is calculated for each window. This pps is compared to the background traffic pps to detect the attack. Out of 23 chosen attacks, 21 are detected using the LoSS technique. It is important to note that such a technique is useful only if there is a very high intensity of pps during the attack.

It can be noted that applications of such methods to anomaly detection in network security is limited to only some types of attacks in which there is a significant change in the nature of network feature being monitored. DoS flooding and certain forms of port scans are detectable due to the inherent anomalous alteration generated in patterns of activities. Nevertheless, low-frequency scans and other forms of DoS attacks do not generate such patterns, though their behavior is anomalous. In such cases, these approaches may not perform well.

2.8 Some Open Problems

Intrusion detection systems have the potential to alleviate many of the problems of modern computer networks. A large amount of research has been done and is being done to improve the performance of IDS in terms of accuracy, timely-detection, learning new attacks etc. While various techniques from diversified areas like machine learning, bio-computing and immune system have been applied to achieve the desired performance, there remains much that needs to be addressed and researched in this arena. After surveying many articles including review and survey articles, we outline some of the open problems in this area that require further investigation and research.

I. What is the accepted rate of false alarms?

II. What are the idea feature sets for different data mining techniques?

III. Beside system calls and network data, is there any other thing to monitor which is more fundamental for learning attacks or normal behavior?

IV. How much data is required in order to properly train a data mining technique?

V. Is there any well-defined way to combine various specialized techniques to have a more robust IDS?

VI. How true is the assumption that employing data mining techniques on a network other than the one on which they were trained renders them ineffective, or at least seriously degrades their performance?

VII. Is it possible to characterize different attacks under one umbrella using some mathematical modelling techniques or by some other techniques?

VIII. Should normal usage profiles be based on individual hosts and services, or should hosts and/or services be grouped together?

IX. Is it possible to arrive at a solution which, when applied or followed, will make, at least one type of attacks, ineffective?

2.9 Conclusions

In this chapter, we present a survey on existing techniques on IDS. Over the last decade, there has been a boom in the research on IDS. Researchers from various branches of computer science have contributed a lot to improve the performance of IDS. There have been efforts to learn a stable behavior of normal by monitoring system calls. Such approaches are inspired by human immune system. Various data mining and machine learning techniques have shown a very good performance as far as the accuracy of the IDS is concerned, for example classification and clustering methods. Data mining methods have also been used to make intrusion detection an automatic process, with a minimum human interference. Such methods are used to extract signatures of normal as well as abnormal activities. There is lobby of researchers who are trying to take IDS as a process of network traffic monitoring and applying various traffic analysis techniques, e.g statistical based methods and wavelets, to build an IDS. These methods assume network data as time series and make use of some characteristics of network traffic, e.g. self-similarity. Most of these methods are confined to detecting DoS attacks, where there is a sudden change/burst in the network activities.

The most of the proposed methods are concentrating on algorithms to detect or differentiate attacks from normal data. There has been a very little efforts to modify proposed algorithms and methods to make attack detection process faster. The problem of high false alarms is seldom taken as the focus of the proposed research work. The above noted points may be some of the causes for the gap between commercial products and research proposals. As pointed out in the section 2.8, a very low false positive rate and nearly real-time detection should be the focus of future research in IDS. Artificial immune system based approaches have yet to contribute enough to make *computer immune system* as robust as *human immune system*.

The described work in the chapter is published as:
Sanjay Rawat , Arun K. Pujari, V. P. Gulati (2005) "Intrusion Detection Systems: A Survey on Algorithms and Methods." Technical Report. Submitted to Journal.

CHAPTER III

APPLICATION OF SINGULAR VALUE DECOMPOSITION TO FAST INTRUSION DETECTION

"...it doesn't matter how beautiful your theory is, it doesn't matter how smart you are – if it doesn't agree with experiment, it's wrong."

- Richard P. Feynman

3.1 Introduction

This chapter presents a method to reduce the dimensionality of the data, to be analyzed by the IDS, without degrading the performance of the IDS. Such a method is useful in making an IDS fast to detect the attack early.

The increasing frequency and complexity of Internet attacks has raised the level and volume of knowledge required by systems administrators to effectively detect the intrusions. Intrusion detection systems collect information from network activities and system activities. Anomaly detection systems attempt to model (learn) usual or normal behavior, defined by the security policy of the organization. Any significant deviation from this established behavior is flagged as a potential intrusion. Anomaly based systems face a problem of getting clean normal data for its learning. Often in live network, it is very difficult to claim to have a clean normal data. Normally the data contains some background activities, termed as noise, which some times degrades the overall performance of the ID system. Also, as mentioned earlier, a good amount of knowledge is required to detect the presence of the attack. In other words, we should collect as much information as possible so that attack can be manifested in this, but larger the amount of data to process, higher will be the cost in terms of time and space, associated with it. This leads to a delay in identifying the presence of attack and thus defeats the main purpose of having IDS i.e. timely detection of the attack. Present breed of networks works on MBPS and GBPS, thereby producing a huge amount of data in seconds. This imposes a challenge for modern ID systems to process data at this speed. Also, our experience with DARPA data shows that it is very difficult to extract normal and significant process-level information without getting some other unnecessary information as, temporally, data is located very dense. Thus we may get some information, which is not useful in distinguishing normal processes from abnormal ones. In the view of above discussion, *practically it is desirable to process as less data as possible which adequately captures all the information required for the manifestation of attacks*. Also it has been observed [88] that Data Mining based research in IDS concentrates on the construction of operational IDSs, rather than on the discovery of new and fundamental insights into the nature of data, attacks and false positives. The main emphasis is on data mining steps, and other KDD (Knowledge Discovery in Databases) are largely ignored. The observations, mentioned above, set the motivation for our work. In the present work, we focus on preprocessing of the data to be analyzed for intrusion detection and investigate a method of reducing the dimensionality of the data being fed into the host-based system without degrading the performance of the system, by showing rigorous experimental analysis. This work borrows ideas from one of the methods of information retrieval - *Latent Semantic Indexing* (LSI) [41][17]. We make use of Spectral Analysis to reduce the size of the vectors (explained later) derived from the sequence of system calls by using *Singular Value Decomposition* (SVD) [67].

The rest of the chapter is organized as follows. Section 3.2 covers topics which serve as the background work for the proposed method. This includes *latent semantic indexing*, *singular value decomposition*, *cosine metric* and *ROC curve* and *AUC score*. Section 3.3 describes the method to represent a process as vector. In section 3.4, we provide the justification of using SVD, by showing some empirical results and present a kNN based scheme for intrusion detection. The experimental results on DARPA'98 data and UNM data are shown in section 3.5, which is followed by the conclusions in section 3.6

3.2 Preliminary Background

In this section, we provide necessary background to understand the technique, used in our scheme. According to Denning, normal behavior of entities is substantially different from the intrusive behavior (abnormal behavior). Therefore, in case of process monitoring approach, under the normal execution a process exhibits a pattern, which is different from the pattern under an effort to exploit the process to launch an attack. A process can be presented in terms of the system calls made by it. It should be noted that in order to exhibit a specific pattern under normal condition, there should be some high correlation among the system calls made by the process. In other words, occurrence of some system calls leads to the presence of other related system calls. If this co-occurrence of system calls can be captured, the detection of the abnormal execution of the process becomes easier task. In this regards, we try to investigate the idea of applying Latent Semantic Indexing (LSI) method to build an anomaly based IDS.

3.2.1 Latent Semantic Indexing/Analysis

Latent Semantic Indexing (LSI) is a relevant documents retrieval method [41], being used by search engines [17]. In addition to learn which keywords a document contains, the method examines the document collection as a whole, to see which other documents contain some of those same words. LSI considers documents that have many words in common to be semantically close and ones with few words in common to be semantically distant. Natural language is full of redundancies, and not every word that appears in a document carries semantic meaning. In fact, the most frequently used words in English are words that don't carry *content* at all, e.g. functional words, conjunctions, prepositions, auxilliary verbs and others. LSI looks at patterns of words distribution (specifically, word co-occurrence) across a set of documents. In the process of applying LSI, a word-by-document matrix $A = [a_{ij}]$ is constructed where a_{ij} is the frequency of i^{th} word in the j^{th} document, $i = 1, ..., m$ and $j = 1, ..., n$. Thus each document is represented as a vector in m-dimensional term-space [41]. In this space, documents that have many words in common will have vectors that are near to each other, while documents with few shared words will have vectors that are far apart. LSI works by projecting this large, multidimensional space down into a smaller number of dimensions by decomposing the matrix A using a Linear Algebraic method called Singular Value Decomposition (SVD), which is described in detailed in the next section. In doing so, words that are semantically similar will get squeezed together and will no longer be completely distinct.

3.2.2 Singular Value Decomposition

SVD is closely related to a number of mathematical and statistical techniques in a wide variety of other fields, including eigenvalue decomposition, spectral analysis, and fractal analysis [67]. Let A be the term-by-document matrix as described above. Then SVD of A is given by

$$A = T_0 S_0 D_0^{'} \tag{12}$$

where T_0 and D_0 are the matrices of left and right singular vectors and S_0 is the diagonal matrix of singular values (square roots of eigenvalues of $A^T A$ or AA^T) in decreasing order, of A. T_0 and D_0 have orthonormal columns. These eigenvalues describe the variance of each of the components. Initially S_0 is a $n \times n$ matrix with singular values filling the first r places on the diagonal, $r = rank(A)$. SVD allows a simple strategy for optimal approximate fit using smaller matrices. In the matrix S_0, the first l largest values are kept and accordingly the size of matrices T_0 and D_0 are adjusted as follow:

$$A_l = T_l S_l D_l^{'} \tag{13}$$

where T_l is $m \times l$ matrix, S_l is $l \times l$ matrix and $D_l^{'}$ is $l \times n$ matrix. The new matrix A_l is approximately equal to A and is of rank l. In some sense, SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors, whereby each term and document is represented by a vector in $l - space$ using elements of the left or right singular vectors. The amount of dimension reduction i.e. the choice of l is critical to our work and should be chosen carefully. Any similarity measure (e.g. cosine similarity) or dot product between two column vectors of the product $S_l D_l^{'}$ reveals the extent to which two documents have a similar profile of terms.

The SVD of $A_{m\times n}$ can also be represented as follows:

$$A = T_0 S_0 D_0^{'} = \sum_{k=1}^{p} s_k t_k d_k^T \tag{14}$$

where $p = min(m, n)$. The following properties of SVD make it suitable for applications where the notion of distance is involved [**?**].

Property 1: If A is an $m \times n$ matrix with rank r with SVD $A = T_0 S_0 D_0^{'}$, then the Euclidean distance between any two column vectors of A is equal to the weighted Euclidean distance between the corresponding columns of V^T, where the weighting is by the singular values s_k, i.e.

$$||a_i - a_j||^2 = \sum_{k=1}^{r} s_k^2 (d_{ik} - d_{jk})^2 \tag{15}$$

Property 2: Let A be an $m \times n$ matrix with rank r with SVD $A = T_0 S_0 D_0^{'}$, and $V^T = [\phi_1, \phi_2, \ldots, \phi_n]$. If two columns of A are the same, $a_i = a_j$, then the first r elements of the ith and jth columns of V^T are equal, i.e. $\phi_i = \phi_j$. This property can be generalized to state that if two columns of A are proportional, i.e., $a_i = \alpha a_j$, then the corresponding columns of V^T are proportional by the same constant upto the rank of A, i.e.,

$$\phi_{ki} = \alpha \phi kj \;\; k = 1, 2, \ldots, r.$$

Any query q is also treated as a pseudo-document, where q is a row vector of size $l \times m$. This new vector is mapped into the new space as follows:

$$\hat{q} = q T_l S_l^{-1} \tag{16}$$

The sum of these l-dimensional terms vectors is reflected by the qT_l term in the above equation and the right multiplication by S_l^{-1} differentially weights the separate dimensions. Thus, the query vector is located at the weighted sum of its constituent term vectors. This transformed query can be treated as one of the columns of matrix A_l, and thus can be compared against rest of the columns to find most similar documents, as mentioned above.

3.2.3 Cosine Similarity Metric

Let there be two vectors P_i and P_j of the same length (dimension). The cosine similarity metric $CosSim(P_i, P_j)$ measures the similarity between two processes in terms of the angle between the two processes and is defined as follows:

$$CosSim(P_i, P_j) = \frac{P_i \cdot P_j}{\parallel P_i \parallel \cdot \parallel P_j \parallel} \tag{17}$$

where $\parallel X \parallel = \sqrt{X \cdot X}$.

In the next section, we describe two techniques to evaluate the performance of the algorithms.

3.2.4 Receiver Operating Characteristic (ROC) Curve and Area Under (ROC) Curve (AUC)

ROC curve is a 2-dimensional plot for classification performance, with *true positive rate* (TP) on the y-axis and *false positive rate* (FP) on x-axis. Each point in ROC space is a single confusion matrix. An ROC curve is formed from a sequence of such points, connected by line segments. In most of the cases, the sequence of points is generated by changing the threshold for the *distance* (or *similarity*) metric, being used for classification. However, it may be noted that given the ROC curves for two classifiers, it is very difficult to compare the performance of two classifiers. The only method is to visually inspect both the graphs. Also, by definition, ROC curves are very sensitive to the choice of threshold. Finer the incremental steps for threshold, more optimal combination of TP and FP is expected to come. In the process of classifying, most of the classifiers yield a ranking of examples. But ROC curve are insensitive to ranking of examples. To make the point clearer, let us consider the following example. Table 9 shows the results of two classifiers, classifier1 and classi-

Classifier	Actual -	Actual +
classifier1	- - - - +	- + + + +
classifier2	+ - - - -	+ + + + -

Table 9: Classification results of two classifiers

fier2, for a set of 10 testing examples, arranged in the increasing order of being positive.

Clearly, both classifiers produce an accuracy of 80%, and therefore, the two classifiers are equivalent in terms of accuracy. However, intuition tells us that classifier1 is better than classifier2, as overall positive examples are ranked higher in classifier1 than classifier2. To overcome such shortcomings, a better measure of performance, termed as AUC, is used [74].

The AUC is defined in terms of ROC curve. Let $\hat{p}(x)$ be the estimate of the probability that as object with measurement vector x belongs to class 0. Let$f(\hat{p}) = f(\hat{p}(x)|0)$ be the probability function of the estimated probability of belonging to class 0 for class 0 points, and let $g(\hat{p}) = g(\hat{p}(x)|1)$ be the probability function of the estimated probability of belonging to class 1 points. Let $F(\hat{p}) = F(\hat{p}(x)|0)$ and $G(\hat{p}) = G(\hat{p}(x)|1)$ be the corresponding cumulative distribution functions. The ROC curve is defined as a plot of $G(\hat{p})$, on vertical axis, against $F(\hat{p})$, on horizontal axis. Clearly this plot lies in a unit square. A good classification rule is reflected by an ROC curve which lies in the upper left triangle of the square. The AUC is then simply the area under the ROC curve. The definition of distribution function says that a distribution function $D(x)$ describes the probability that a variate X takes on values less than or equal to a number x. Therefore, for an arbitrary point $\hat{p}(x) = t$, the probability that a randomly chosen class 1 point will have a $\hat{p}(x)$ smaller than t is $G(x)$. Suppose that t is chosen randomly according to the distribution F. Then the probability that the randomly chosen class 1 point will have a smaller value of $\hat{p}(x)$ than the randomly chosen class 0 point is

$$\int G(u)f(u)du \tag{18}$$

However, from the definition of ROC, the AUC is given by:

$$\int G(u)dF(u)du = \int G(u)f(u) \tag{19}$$

From the equations (18) and (19), it can be asserted that the AUC is equivalent to the probability that a randomly chosen member of class 1 will have a smaller estimated probability of belonging to class 0 than a randomly chosen member of class 0.

To obtain a working formula for AUC, let $f_i = \hat{p}(x_i)$ be the probability of belonging to class 0 for the ith class 0 point from the test set, $i = 1, \ldots, n_0$ (n_0 is the number of positive examples). Similarly, let us define $g_i = \hat{p}(x_i)$ for n_1 test points belonging to class 1. Then $\{g_i\}$ and $\{f_j\}$ are samples from g and f distributions respectively. After ranking them in the ascending order of probabilities being positive, let r_i be the rank of ith element for class 0. Then there are $(r_i - i)$ class 1 points with probabilities of belonging to class 0, smaller than the ith point. Therefore, total number of such pairs (one point from class 0 and one from class 1) such that class 1 point has smaller probability of belonging to class 0 than class 0 points is:

$$\begin{aligned} \sum_{i=1}^{n_0}(r_i - i) &= \sum r_i - \sum i \\ &= S_0 - n_0\tfrac{(n_0+1)}{2} \end{aligned} \tag{20}$$

where S_0 is the sum of the ranks of the class 0 test points. Since there are n_0n_1 such pairs of points altogether, our estimate of the probability that a randomly chosen class 1 point has a lower estimated probability of belonging to class 0 than a randomly chosen class 0 point is

$$\hat{A} = \frac{S_0 - n_0\frac{(n_0+1)}{2}}{n_0n_1} \tag{21}$$

The important point to note here is that no mention of a threshold has been made. The measure $\hat{A}$ is an overall measure of how well separated are the estimated distributions of $\hat{p}(x)$ for class 0 and class 1.

Using the formula given by equation (21) and example given in table 9, we get $\hat{A}_1 = \frac{24}{25}$ for the classifier1 and $\hat{A}_2 = \frac{16}{25}$ for the classifier2. Thus AUC is indeed a better performance indicator than ROC.

3.3 *Vector Representation of Processes in Terms of System Calls*

Let S (say, $Card(S) = m$) be a set of system calls made by all the processes under normal execution. From all the normal processes a matrix $A = [a_{ij}]$ is formed, where a_{ij} denotes the frequency of i^{th} system call in the j^{th} process. We also form a matrix $B = [b_{ij}]$ where, $b_{ij} = 1$, if i^{th} system calls is present in the j^{th} process, otherwise $b_{ij} = 0$. Thus the binary representation of process P, namely Pb_j, is defined by the $m - vector\ Pb_j = [0, 1]^m$ as a column in B (For continuity, we have introduced the construction of matrix B here, but we will make use of B in the next chapter). For example, Let S = {`access audit chdir close creat exit fork ioctl`}.

Let the two processes be

P_1 = `access close ioctl access exit`, and

P_2 = `ioctl audit chdir chdir access`. Then we have:

$$A = \begin{array}{c} \begin{array}{cc} P_1 & P_2 \end{array} \\ \begin{pmatrix} 2 & 1 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \end{array} \qquad B = \begin{array}{c} \begin{array}{cc} Pb_1 & Pb_2 \end{array} \\ \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \end{array}$$

The rows of A (and B) correspond to the elements of S in the same order and columns of A (and B) correspond to processes P_1 and P_2. Thus the first entry in A is calculated by counting the frequency of system call `access` in the process P_1 that is 2. Similarly the first entry of the second column of A is calculated by counting the frequency of the system calls `access` in the process P_2 which is 1, and so on. Similarly the first entry of the first column of B is 1 because the system call `access` is present in P_1 whereas the second entry is 0, which shows that the system call `audit` is absent in P_1.

3.4 *Singular Value Decomposition for Feature Extraction and Refinement*

As noted earlier, processes can be represented as the sequences of system calls made by them. These sequences show patterns in terms of co-occurrence of system calls under normal execution. Once the incidence matrix is constructed, each process can be treated as a vector of length equal to the number of unique system calls. At this point, we would like to raise a question that *is it possible to reduce the size of each vector i.e. to reduce the size of incidence matrix without loosing relevant information contained in the matrix?* Also, in order to classify a process as either normal or abnormal, we make use of kNN classifier (the algorithm is described later in this section), but any other classifier, which works on vector model, can be used. But kNN classifier suffers from what is referred to as *curse of dimensionality* in the literature. The issue is that the similarity (or distance) between two instances depends upon all the attributes i.e. if there are n attributes, then the similarity metric depends on all the n dimensions in Euclidean space. But it may well be a case that not all the n attributes are useful in determining the similarity, but only ,say, r attributes are relevant. Thus the remaining $n - r$ attributes distort the similarity score and bring in some error. One approach to overcome this problem is to weight each attribute differently when calculating the similarity between two instances. This is equivalent to stretching the axes, corresponding to relevant attributes, in the Euclidean space and shortening the axes that correspond to less relevant attributes. For this reason, we make use of a spectral analysis technique- SVD, in the stage of preprocessing of the data to reduce the dimensionality of the space. The action of singular values on the incidence matrix is to stretch or shorten the axes corresponding to attributes according to their relevance (see equation (15)). Higher singular values corresponds to more relevant attributes whereas lower singular values corresponds to less relevant attributes. Therefore the truncated form, denoted as A_l, of A is calculated by using equation (13). The processes in the reduced dimensional space corresponds to the columns of the product $S_l D_l^{'} = A_{reduced}$, say. The main advantage of using SVD is that it tries to capture system calls which occur together in most of the processes and projects them closer into a new space of lower dimension. In this way, processes sharing the common system calls will become more similar than those sharing very few system calls in common. In context of IDS, every new process is considered as a query to be classified. Thus each new process P is first mapped into the new space by using equation (16) to get reduced vector $\hat{P}$. Thus, we have the reduced form of original incidence matrix A, i.e. $A_{reduced}$ and the reduced query vector $\hat{P}$ of P.

3.4.1 Information Containment: What is Removed by SVD

In the new lower dimensional space, only the most relevant feature are reflected by means of new axes, which are further controlled (or scaled) by singular values. In the parlance of system calls based IDS, very small singular values correspond to less important system calls, thereby making their effect very less or zero, if we discard those singular values. For example, in the DARPA dataset that we use for experimentation, it is found the *mmap* system call appears in almost all the processes with high frequency. The mmap system call is used to map length bytes starting at offset from the file (or other object) specified by the file descriptor into memory. The actual place where the object is mapped is returned by mmap. On success, mmap returns a pointer to the mapped area. Whenever a function is called in the program, mmap is also used. Therefore the presence or absence of this system call does not infer any knowledge about the nature of processes. To see experimentally that SVD indeed takes care of such system calls, we calculate incidence matrices with and without mmap system calls. After selecting the 12 largest singular values, we find that those are almost same (see table 10), which shows that any large singular value does not correspond to mmap system calls. Another point to mention is that if mmap does not affect

with mmap ($\times$1000)	4.1671 3.9450 1.3133 0.9367 0.8348 0.5180 0.4798 0.3985 0.3147 0.1946 0.1410 0.1021
without mmap ($\times$1000)	4.1518 3.9368 1.3025 0.9363 0.8275 0.5101 0.4113 0.3450 0.2197 0.1841 0.1407 0.1017

Table 10: Singular values in the presence and absence of mmap system. It can be observed that singular values are almost same.

the nature of process, then the coordinates of any process, in the reduced dimensional space, should not change much. In other words, mutual similarity among the processes should be unaffected by the mmap system calls. Figure 8 and 9 depict the cases of dot product among the processes in the presence and absence of mmap, respectively. Both of the figures represent the same behavior of processes. [h]

Therefore, we have shown empirically that SVD indeed removes irrelevant system calls. This not only reduces the computational time but also removes noise, as afore mentioned and is shown by means of experimental results in the next section.

3.4.2 kNN classifier for Intrusion Detection

In order to testify that in its reduced form $A_{reduced}$, the incidence matrix A contains all the relevant information to discriminate between normal and abnormal process, we choose

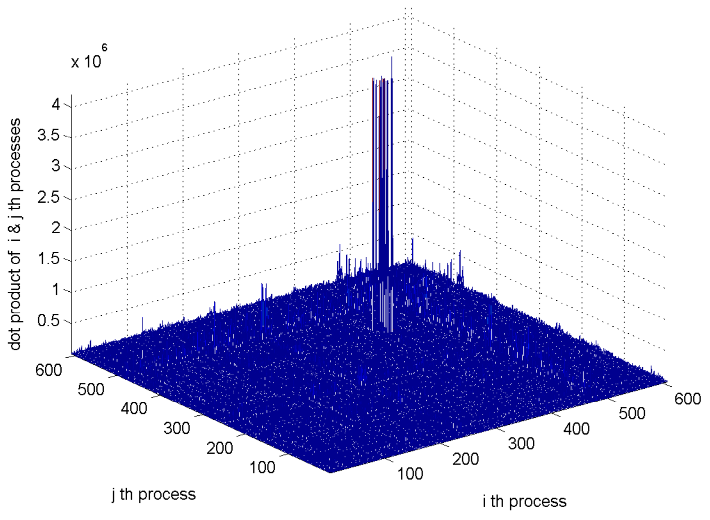

Figure 8: Values of dot product between processes in the presence of mmap system calls.

kNN classifier as a classification algorithm (more details on kNN are provided in the next chapter). However, we make it clear that any other vector model-based classification algorithm can be used. kNN algorithm has already been shown to perform well for this task by Liao and Vemuri [120]. We calculate $A_{reduced}$ and the new process $\hat{P}$. This new transformed process $\hat{P}$, is matched against the columns of $A_{reduced}$ using the cosine similarity measure given by equation (17). At this stage kNN-algorithm is applied to classify the process as either normal or abnormal. In other words, cosine similarity score of $\hat{P}$ with each column of $S_l D_l'$ is calculated and sorted in descending order. The first k scores are selected and their average is calculated. If this average is greater than some predefined threshold value, then the process is declared as normal otherwise abnormal. The pseudo-code of the algorithm is shown in the figure 10.

3.5 Experimental Setup and Results

Our objective is to show the use of SVD to reduce the dimension of the space and remove the noise without degrading the performance of the system. We choose two data sets to perform the experiments- DARPA'98 BSM data and UNM data for various applications as sendmail, inetd, login and ps. The code for performing all the experiments is written in

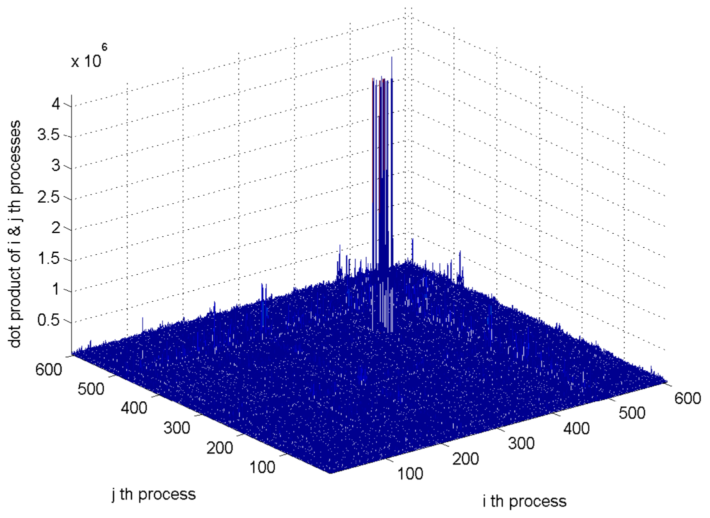

Figure 9: Values of dot product between processes in the absence of mmap system calls.

Matlab [137], running on a Windows 2000 machine.

3.5.1 Experiments with DARPA data

DARPA'98 BSM data [35] is widely used data set for the evaluation of host-based IDS. We use BSM audit logs from the 1998 DARPA data for training and testing of our algorithm. As we wish to compare our proposed method with that of Liao and Vemuri, we use the same data set that is used in their scheme.

After analyzing the training data, we extract the 50 unique system calls that appear in the training data. All the 50 system calls are shown in table 11. For each day of data, a separate BSM file is provided with the 'BSM List File'. Each line of this file contains the information about one session such as time, service, source IP and destination IP. A '0' at the end of the line shows that the session is normal and the presence of a '1' at the end of the line declares the session intrusive. All the intrusive sessions are labeled with the name of the attacks launched during the sessions. We make use of the BSM commands auditreduce and praudit and a couple of shell scripts to extract the data that can be used in our algorithm. On analyzing the entire set of BSM logs (list files), we locate the five days which are free of any type of attacks - Tuesday of the third week, Thursday of the fifth

```
Given a set of normal processes and a set of system calls $S$, form the matrix $A = [a_{ij}]$
Calculate $A_l = T_l S_l D_l^{'}$ using first $l$ largest singular values;
Calculate the product $A_{reduced} = S_l D_l^{'}$;
for each process $P$ in the test data do
    if $P$ has some system calls which does not belongs to $S$ then
        P is abnormal; exit;
    else
        calculate $\hat{P} = P T_l S_l^{-1}$
        for each process $A_j$ in the training data $A_{reduced}$ do
            calculate $CosSim(\hat{P}, A_j)$;
            if $CosSim(\hat{P}, A_j)$ equals 1.0 then
                P is normal; exit;
        end do
        find first k highest values of $CosSim(\hat{P}, A_j)$;
        calculate $Avg\_Sim$ for $k$ nearest neighbors so obtained;
        if $Avg\_Sim$ is greater than $Sim\_Threshold$ then
            P is normal;
        else P is abnormal;
end do
```

Figure 10: Pseudo code for the proposed SVD-based Scheme

```
access, audit, auditon, chdir, chmod, chown, close, creat,
execve, exit, fchdir, fchown, fcntl, fork, fork1, getaudit,
getmsg, ioctl, kill, link, login, logout, lstat, memcntl,
mkdir, mmap, munmap, nice, open, pathconf, pipe, putmsg,
readlink, rename, rmdir, setaudit, setegid, seteuid, setgid,
setgroups, setpgrp, setrlimit, setuid, stat, statvfs, su,
sysinfo, unlink, utime, vfork
```

Table 11: List of 50 unique system calls

week and Monday, Tuesday and Wednesday of the seventh week. We choose the first four days for our training data and the fifth one for the testing of the normal data to determine the false positive rate. There are around 2000 normal sessions reported in the four days of data. We extract the processes occurring during these days and our training data set consists of 606 unique processes. There are 412 normal sessions on the fifth day and we extract 5285 normal processes from these sessions. We use these 5285 normal processes for the testing data. In order to test the detection capability of our method, we incorporate 55 intrusive sessions into our testing data. Table 12 lists these attacks. A number in the beginning of the name denotes the week and day and the later part denotes the name of the session (attack). For example, the attack name 3.1_it_ffb_clear means that the attack was launched in the 3^{rd} week, on the 1^{st} day viz. Monday and the name of the attack is ffb (in clear mode). The table should be read horizontally from left to right and top to bottom. These attack sessions consist of almost all types of attacks launched on the victim

1.1_it_ffb_clear, 1.1_it_format_clear, 2.2_it_ipsweep, 2.5_it_ftpwrite, 2.5_it_ftpwrite_test, 3.1_it_ffb_clear, 3.3_it_ftpwrite, 3.3_it_ftpwrite_test, 3.4_it_warez, 3.5_it_warezmaster, 4.1_it_080520warezclient, 4.2_it_080511warezclient, 4.2_it_153736spy, 4.2_it_153736spy_test, 4.2_it_153812spy, 4.4_it_080514warezclient, 4.4_it_080514warezclient_test, 4.4_it_175320warezclient, 4.4_it_180326warezclient, 4.4_it_180955warezclient, 4.4_it_181945warezclient, 4.5_it_092212ffb, 4.5_it_141011loadmodule, 4.5_it_162228loadmodule, 4.5_it_174726loadmodule, 4.5_it_format, 5.1_it_141020ffb, 5.1_it_174729ffb_exec, 5.1_it_format, 5.2_it_144308eject_clear, 5.2_it_163909eject_clear, 5.3_it_eject_steal, 5.5_it_eject, 5.5_it_fdformat, 5.5_it_fdformat_chmod, 6.4_it_090647ffb, 6.4_it_093203eject, 6.4_it_095046eject, 6.4_it_100014eject, 6.4_it_122156eject, 6.4_it_144331ffb, test.1.2_format, test.1.2_format2, test.1.3_eject, test.1.3_httptunnel, test.1.4_eject, test.1.5_processtable, test.2.1_111516ffb, test.2.1_format, test.2.2_xsnoop, test.2.3_ps, test.2.3_ps_b, test.2.5_ftpwrite, test.2.4_eject_a, test.2.2_format1

Table 12: List of 55 attacks used in testing data set

Solaris machine (in the simulated DARPA setup) during seven weeks of training period and two weeks of testing period and that can be detected using BSM logs. An intrusive session is said to be detected if any of the processes associated with this session is classified as abnormal. Thus detection rate is defined as the number of intrusive sessions detected, divided by the total number of intrusive sessions.

With the statistics, given above, the incident matrix A is of the size 50×606 . We calculate the SVD of A. To reduce the dimension of the matrix, we choose the singular values greater than 100 ($l = 12$), greater than 20 ($l = 23$) and greater than 10 ($l = 31$) for experimentation. For kNN classifier, we choose three values $k = 5, 10, 15$. To measure the effectiveness of the approach, we perform experiments with different combinations of these values. It is observed that results for $k = 10$ and $k = 15$ are almost same. Therefore, we omit the results for $k = 15$. Tables 13, 14 and 15 summarize the results for $l = 12, 23, 31$ at $k = 5, 10$. The first column in the table shows the threshold values used in the experiments. Entries in column two are the rate of false positives, which is equal to the number of normal processes detected as abnormal divided by the total number of normal processes. Column three details the detection rate as defined above.

$l = 12$				
Threshold	$k = 5$		$k = 10$	
-	FP	DP	FP	DR
0.35	0.000000	0.345455	0.000000	0.345455
0.40	0.000000	0.345455	0.001135	0.745455
0.45	0.000189	0.345455	0.001135	0.745455
0.50	0.000189	0.400000	0.002838	0.745455
0.55	0.001703	0.745455	0.004730	0.745455
0.60	0.007758	0.763636	0.010596	0.781818
0.64	0.022895	0.781818	0.061684	0.800000
0.68	0.037086	0.781818	0.126206	0.909091
0.71	0.088742	0.909091	0.129423	0.909091
0.74	0.105771	0.945455	0.140587	0.945455
0.78	0.156291	0.981818	0.183160	1.000000
0.81	0.233869	1.000000	-	-

Table 13: False Positive Rate vs Detection Rate for different values of $K = 5, 10$ at $l = 12$

$l = 23$				
Threshold	$k = 5$		$k = 10$	
-	FP	DP	FP	DR
0.15	0.000000	0.454545	0.000000	0.454545
0.20	0.001703	0.454545	0.003027	0.472727
0.25	0.003406	0.545455	0.027058	0.581818
0.30	0.060927	0.600000	0.069253	0.618182
0.34	0.090823	0.654545	0.125071	0.945455
0.38	0.176727	0.981818	0.209461	0.981818
0.42	0.217786	0.981818	0.229518	0.981818
0.46	0.259224	1.000000	0.262441	1.000000

Table 14: False Positive Rate vs Detection Rate for different values of $K = 5, 10$ at $l = 23$

At $l = 21$ and $k = 10$, we achieve a 100% detection rate at 18.31% false positive rate (table 13), which is quite a comparable result with other schemes. The worst case is 100% detection rate at 26.24% false alarms at $l = 23$ and $k = 10$.

We calculate results using Liao's scheme for comparison to show the usefulness of our approach. Following table 16 summarizes the findings with Liao's scheme.

In this case, we attain a detection rate of 100% at 39% false positive with vector size 1×50 , whereas it is 100% detection rate at 18.31% false positive rate with vector size 1×12 . Thus, we achieve better performance with vector size as low as one third of the vector size in Liao's scheme. The total computing time in Liao's scheme is about **1900** seconds, whereas it is **1500** seconds in our method. We consider this as a good improvement as far

$l = 31$				
Threshold	$k = 5$		$k = 10$	
-	FP	DP	FP	DR
0.08	0.000000	0.345455	0.000189	0.345455
0.10	0.000189	0.345455	0.000757	0.381818
0.15	0.035005	0.672727	0.037275	0.672727
0.20	0.071145	0.709091	0.086282	0.727273
0.24	0.085525	0.963636	0.155535	0.963636
0.28	0.143425	0.981818	0.193377	0.981818
0.31	0.217597	1.000000	0.248061	1.000000

Table 15: False Positive Rate vs Detection Rate for different values of $K = 5, 10$ at $l = 31$

Threshold	FP	DR
0.50	0.00	0.34
0.55	0.0009	0.34
0.70	0.001	0.36
0.78	0.001	0.74
0.85	0.002	0.74
0.88	0.031	0.76
0.90	0.034	0.78
0.93	0.035	0.81
0.95	0.035	0.83
0.97	0.044	0.92
0.98	0.091	0.96
0.99	0.33	0.96
0.992	0.36	0.98
0.994	0.39	1.00

Table 16: False Positive Rate vs Detection Rate for k= 10 for Liao and Vemuri scheme

as the computing time and timely response is concerned.

Another thing which is to be noted is that in table 15, the threshold value at which DR is 100% is only 0.31, which is very low as compared to that in Liao's scheme i.e. 0.994. It shows that SVD-based scheme can widen the difference between normal and abnormal processes by removing the unnecessary stuff as noise. The effect of different combinations of number singular values l and values of k can be seen in ROC curves shown in figures 11 and 12.

A different perspective of looking at detection rate and false alarm is "what is the detection rate at 0% false alarm or 1.0% false alarms and what is the false alarm rate at 90% or 100% detection rate?" Such type of statistics is given in table 17. Entries in the bold indicate the values which are fixed to observe the outcome against them.

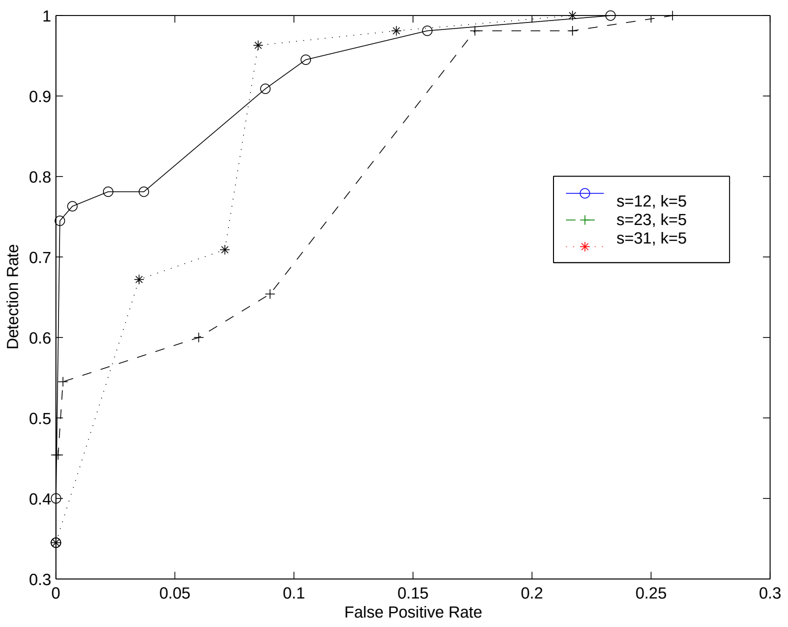

Figure 11: Variation in ROC curves for $l = 12, 23, 31$ at $k = 5$.

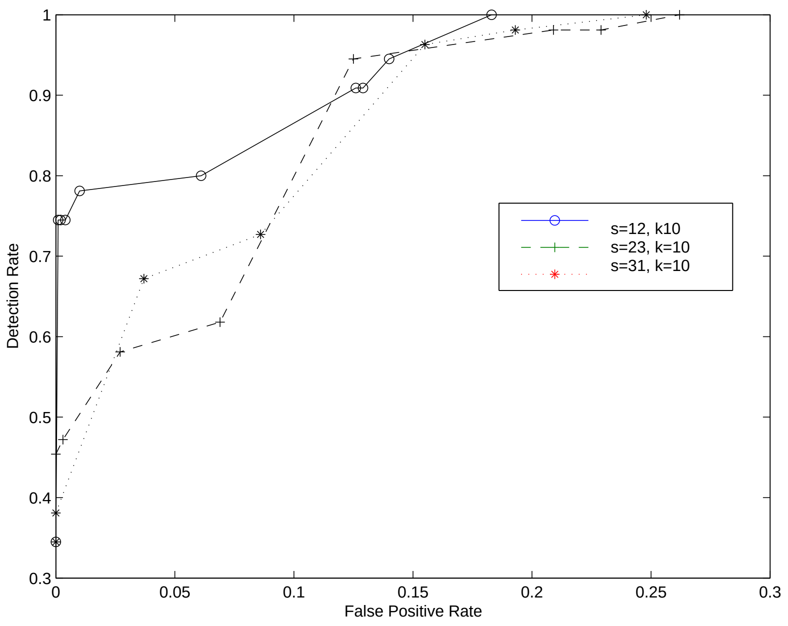

Figure 12: Variation in ROC curves for $l = 12, 23, 31$ at $k = 10$.

l=12				*l*=23				*l*=31			
k=5		*k*=10		*k*=5		*k*=10		*k*=5		*k*=10	
FP	DR	FP	DR	FP	DR	FP	DR	FP	DR	FP	DR
0.0	74.5	**0.0**	74.5	**0.0**	45.5	**0.0**	47.2	**0.0**	34.5	**0.0**	38.1
1.0	76.36	**1.0**	78.18	**1.0**	54.5	**1.0**	47.2	**1.0**	34.5	**1.0**	38.1
8.8	**90**	12.6	**90**	17.6	**90**	12.5	**90**	8.5	**90**	15.5	**90**
23.3	**100**	18.3	**100**	25.9	**100**	26.2	**100**	21.7	**100**	24.8	**100**

Table 17: False Positive Rate vs Detection Rate for different combinations

3.5.1.1 *Cross-Validation Experiments*

In order to test the accuracy of the classifier for low bias and low variance, we conduct a technique of experimentation, called *ten-fold cross-validation*. In this scheme, the training and testing data are combined and a small part is separated for testing and rest is used for training.The same process is repeated by taking another part for testing and remaining for training in a circular manner. In our set of experimentation, we take 10% of total data for testing and remaining 90% for training. Therefore, we repeat this process 10 times to exhaust all the data in circular manner. The total number of normal unique processes is 676, out of which 609 are used for training and remaining 67 for testing for each round. All the 55 attacks are included in the testing data. The results are summarized in the table 18. Each entry is the aggregation of the outcome of 10 rounds of experiment. The above results

l=12				*l*=24			
k=5		*k*=10		*k*=5		*k*=10	
FP	DR	FP	DR	FP	DR	FP	DR
0.0	47.00	**0.0**	62.60	**0.0**	38.51	**0.0**	37.95
1.0	56.15	**1.0**	70.5	**1.0**	43.59	**1.0**	42.31
21.69	**90.0**	25.48	**90.0**	23.78	**90.0**	21.45	**90.0**
46.86	**100.0**	57.89	**100.0**	32.96	**100.0**	34.45	**100.0**

Table 18: False Positive Rate vs Detection Rate for different combinations under the ten-fold cross validation experiment

(table 18) shows that there is a lot of variation in the data. Therefore, various methods that use the same data may get different results if the choice of the samples of the data, used for training and testing, are not same.

Our next round of experiment involves the data sets obtained from UNM site (http://www.cs.unm.edu/˜immsec/systemcalls.htm) which is described in the next section.

3.5.2 Experiments with UNM Data

Our objective of experimenting with UNM data sets is to test another aspect of LSI, and thus SVD, in the realm of ID systems. As discussed earlier, LSI understands the semantic meaning of the documents and words in the terms of the co-occurrence of words in the documents. This co-occurrence is very high if the documents belong to one category (topic). In case of DARPA BSM data, though the processes, collected for training, are normal, they belong to different applications. One implication of this is that it is very difficult to extracted common meaning components of many different system calls (words) and processes (documents) [41]. This may be a reason that in DARPA data, we get a bit high false positive rate. Data sets at UNM site are available for different applications i.e. each data set belongs to one category (topic). We choose *sendmail*, *inetd*, *login* and *ps* data sets for our experiment. In the following paragraphs, we detail our experimental results for each of them.

3.5.2.1 Sendmail Data Set

The *sendmail* program was traced on an MIT AI lab running SunOS 4.1.1. In normal data set, which is collected for approximately three days, there are originally 71,767 traces. We discard some very large processes and the final set consists of 65026 processes. Out of theses, we keep 3000 processes for testing data set. These remaining 62024 processes are again processed to extract 10764 unique processes. We separate 6000 processes for training and remaining 4764 processes are mixed with already chosen 3000 processes. Thus our normal testing data set consists of 7764 processes out of which at least 4764 are not used in training set. As there is no attack data set with this trace of sendmail, we choose attack data from UNM *synthetic sendmail* data (http://www.cs.unm.edu/$\tilde{\text{i}}$mmsec/data/synth-sm.html) for the following attacks: *Sunsendmailcp* intrusion, *Decode* intrusion, and *Error condition - forwarding loops*. There are a total of eight attacks involved in the testing data set. The incidence matrix A is constructed from the normal processes and decomposed using SVD. For our experiment, we consider singular values greater than 20 ($l = 28$) and first 10 most similar processes for kNN. We get a detection rate of 100% with false positive as low as 0.06%. However, when we analyze the intrusive processes, we find that most of them is containing system calls that are not in the set of normal system calls. That may be due the fact that we choose different data set for intrusive data, in which the mapping may be different. Despite these, we could able to differentiate among normal and abnormal processes with vector size of only 1×28, whereas original size is 1×64.

3.5.2.2 *inetd, login and ps data sets*

There are a total of 3 normal processes for training i.e. for constructing incidence matrix A for *inetd* process. Due to the lack of sufficient number of normal processes [190], we cannot test the false positive rate for this data set. There is one attack against *inetd* program, which involves 29 processes. There is a total of 35 unique system calls made by these 3 processes under normal execution. For our experiment, we choose $l = 3$ and $K = 3$ i.e. we take all the processes into consideration for kNN classifier. At the threshold of 0.25, we could identify the attack. What is most encouraging is that only the vectors of size 1×3 (about 12 times shorter) are sufficient to take decision about normality or abnormality.

For *login* data set, two attacks are recorded, which involve *trojan* intrusion. There are a total of 12 normal processes for training. There are 47 unique system calls occurred during the normal execution of the program. For experiment, we choose the singular values greater than 1 i.e. $l = 6$. We chose $K = 4$ for kNN classifier. At the threshold value of only 0.04, we are able to detect both of the attacks. With this small value, it can be said that all the normal processes should not be detected as intrusions and therefore a very low false positive rate is expected. Once again, we are able to achieve this performance at vector size of 1×6 only (original size is 1×47).

For *ps* data set, two attacks are recorded, which involve *trojan* intrusion. There is a total of 24 normal processes for training and 22 unique system calls. For experiment, we choose the singular values greater than 1 i.e. $l = 6$. We chose $K = 4$ for kNN classifier. At the threshold value of only 0.28, we are able to detect both of the attacks. Once again, we are able to achieve this performance with vector size of only (original size is 1×22).

In the view of above experiments, it can be asserted that data reduction indeed improves the performance of the ID systems by saving computational time and reducing the noise present in the data.

3.6 Conclusions

Instead of presenting a new method of classifying data as normal or abnormal, the present study focuses on the preprocessing of the data. As has been observed by the study of Julisch [88], data mining in intrusion detection is mainly used to construct a ”black box” that detects intrusions. Through this study, we explore a less-paid-attention, yet important area in the process of intrusion detection. Our study shows that by applying LSI technique, dimensionality of the data can be reduced without loosing its performance. It is also established by experimental results that this method of reduction performs better in case of ’per-application’ data set e.g. login, inetd etc. As a future task, we are further analyzing the decomposition of incidence matrix to gain some insight about the association among system calls under normal and abnormal execution to better understand the process profiling. Such knowledge could be used to capture normal profiles of the programs for better intrusion detection systems.

In the above work, we use kNN classifier with cosine metric, after reducing the dimension of the incidence matrix. The cosine metric takes only the frequencies of the system calls, while calculating the similarity. We feel that only frequency may not be a sufficient parameter to discriminate among normal and abnormal processes. In the next chapter, we investigate other parameters to be included in similarity measurement and propose a new similarity metric.

The described work in the chapter is published as:
Rawat S, Gulati V P, Pujari A K. (2004), ”On the Use of Singular Value Decomposition for a Fast Intrusion Detection System” In: Proc of the First International Workshop on Views On Designing Complex Architectures (VODCA 2004), Italy. ENTCS, Elsevier.

CHAPTER IV

AN IMPROVED SIMILARITY METRIC FOR KNN BASED ANOMALY DETECTOR

It is through science that we prove, but through intuition that we discover.

- Henri Poincare.

4.1 Introduction

In this chapter, we propose a new similarity metric to classify process as normal or abnormal by using kNN algorithm.The new metric captures the frequency and commonality of system calls to calculate the similarity.

It has been observed that a normal execution of a process follows a pattern and hence the normal behavior of a process can be profiled by a sequence of system calls. Any deviation in this pattern of system calls is termed an intrusion in the framework of anomaly-based IDS. The problem of intrusion detection thus boils down to detecting anomalous sequence of system calls, which are measurably different from the normal behavior. A team of researchers at University of California, Davis has conducted a series of experiments on anomaly-based IDS using various methods, viz. kNN with cosine metric [120] and Robust Support Vector Machine [79], among others. While the former method adopts a simple approach and is easy to understand, the second method shows better results in terms of false positive rate and detection rate but with the added complexity of using SVMs. The proposed work draws inspiration from the work of Liao and Vemuri [120] on an anomaly-based intrusion detection system. The motivation behind the proposed work is to develop a method that is as simple to understand as the kNN method and yields results that are comparable to those obtained by using RSVM. By means of a series of experiments, we show that the proposed method yields significantly better results than the kNN method with cosine metric. The results are comparable, if not statistically better, with those obtained by the RSVM method.

We propose a new scheme in which we measure the similarity between two processes using a metric that considers two factors - occurrence of system calls shared by the two said processes and the frequency of all system calls in the processes. Due to the way it is constructed, we term this similarity metric Binary Weighted Cosine (BWC) metric.

The rest of the chapter is organized as follows. Section 4.2 provides the necessary background for the work described later in this chapter, which includes description of kNN algorithm and similarity metrics. In section 4.3, the scheme, based on kNN classifier with cosine metric [120], is described and analyzed. Section 4.4 describes a scheme, which makes use of kNN with new improved similarity metric. Experimental results have been shown in the section 4.5. Analysis of the experimental results and a comparative study of BWC, RSVM and kNN with cosine metric are provided in the section 4.6. In section 4.7, we describe the extension of the scheme, presented in the section 4.4, by using Kendall Tau distance to capture the ordering of system calls in a process, which is followed by the experimentation results on DARPA data. We conclude our work in section 4.8.

4.2 *Preliminary Background*

In this section, we provide necessary background to understand the technique, used in our scheme.

4.2.1 k-Nearest Neighbor (kNN) Classifier

kNN classifier is an example of *instance-based* learning. Learning in such algorithms consists of storing the presented training data and when a new instance is encountered, a set of *similar* instances are retrieved and used to classify the new example [143]. The example data are presented to kNN classifier in the form of vectors in n-dimensional space. Most commonly, the nearest neighbors of a new instance are defined in terms of standard Euclidean distance. In nearest-neighbor learning, the target function may be either discrete-valued or real-valued. In case of discrete-valued function $f : R^n \rightarrow V, V = \{v_1, v_2, \dots, v_s\}$, the algorithm is shown in figure 13. The value $\hat{f}(x_q)$ returned by the algorithm as its estimator of target function $f(x_q)$ is just the most common value of f among the k training examples nearest to x_q. To approximate a real-valued target function $f : R^n \rightarrow R$, the 4th

```
Training Phase
   I. for each training example $\langle x, f(x) \rangle$,
        add the example to the list training_examples
Classification Phase
   I. Given a query instance $x_q$ to be classified

  II.   Let $x_1, x_2, \ldots, x_k$ denote the $k$ instances from training_examples that are nearest to $x_q$

 III.   Return $\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
        where $\delta(a,b) = 1$ if $a = b$ and $\delta(a,b) = 0$ otherwise
```

Figure 13: kNN algorithm for approximating a discrete-valued function

line in figure 13 is replaced by

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

It may well be the case that not all the k neighbors of the query point contribute equal to its classification. The points which are similar to the query point should be given more importance. From this standpoint, one improvement to kNN algorithm is to weight the contribution of each of the k neighbord according to their distance to the query point x_q, giving greater weight to closer neighbors. This modified algorithm is called *distance-weighted k-nearest neighbor* algorithm [50]. This can be achieved by modifying the algorithm presented in figure 13 as:

$$\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)), \text{where}$$

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}, \text{d is Euclidean distance}$$

It can be noted that choice of distance metric is very important as different metrics will yield different values and hence, different neighbors. The main goal is to find instances which are similar to the query. Therefore, in the kNN algorithm, similarity metrics, e.g.

cosine metric, can be applied as well. The weighting factor w_i can be a value which adds something more about the similar instances.

The *distance-weighted k- nearest neighbor* algorithm is quite robust to noisy training data and quite effective when it is provided a sufficiently large set of training data. However, this quality also makes it weak in many practical situation wherein it is difficult to have enough data which represents the domain efficiently. Another problem, which this algorithm suffers from, is referred to as *curse of dimensionality* in the literature. The distance between the instances is calculated by taking all the attributes (dimensions) into the consideration. It may happen that not all of the attributes are relevant for similarity search. In fact such irrelevant attributes may lead to erroneous observations. One approach is to discard such attributes as noise and carry out the calculation with the remaining ones. Another interesting approach (also mentioned in the chapter 3) to overcome this problem is to weight each attribute differently when calculating the distance between two instances. This can be achieve by stretching the axes that correspond to more relevant attributes in the Euclidean space and squeezing the axes that correspond to less relevant attributes.

Apart from being very simple to understand, kNN classifier enjoys many advantages over other classifiers. Some of them are:

- Training is very fast
- Learn complex target functions
- Analytically tractable
- Simple implementation
- Nearly optimal in the large sample limit i.e. as $N \to \infty$
- Uses local information, which can yield highly adaptive behavior.

Over a period of time, many variants of original kNN based classifier have been proposed in the literature. The *evidence theoretic* kNN rule [189] is a pattern classification method based on the *Dempster-Shafer theory* of belief functions. In this approach, each neighbor of a pattern, to be classified, is considered as an *item of evidence* supporting certain hypotheses concerning the class membership of that pattern. Based on this evidence, *basic belief masses* are assigned to each subset of the set of classes. Such masses are obtained for each of the k nearest neighbors of the pattern under consideration and aggregated using the *Dempsters rule of combination*.

To overcome the curse of dimensionality, use of *genetic algorithm* is proposed to come up with a variant- *GA/knn* [90]. In this, instead of selecting vector consisting of only 0s and 1s (weights of features), the *genetic algorithm* manipulates a weight vector, in which a discretized real-valued weight is associated with each feature. Prior to kNN classification, the value of each feature is multiplied by the associated weight, resulting in a new set of features which are linearly related to the original ones. The goal of the genetic algorithm is to find a weight vector which minimizes the error rate of the kNN classifier, while reducing as many weights as possible to zero. GA/kNN can efficiently examine noisy, complex, and high-dimensionality datasets to find combinations of features which classify the data more accurately.

Another algorithm, termed as kNNFP (k Nearest Neighbors on Feature Projections), has been proposed to achieve fast classification by storing training instances as their projections on each feature dimension separately [5]. This allows the classification of a new instance to be made much faster than kNN algorithm, since all projected values can be stored in a data structure that allows fast search. The kNNFP algorithm first makes a set of predictions, one for each feature, using kNN algorithm on the projections on a feature. Then, the final classification of an instance is determined through a majority voting on individual classifications made by each feature.

We now define similarity measures, which we use in our scheme to calculate the similarity between processes. The processes are converted into the vectors as described in section 3.3

4.2.2 Binary Similarity Measure

We define a similarity score $\mu(Pb_i, Pb_j)$ between any two processes Pb_i and Pb_j (where $m - vector\ Pb_i = [0, 1]^m$ as a i^{th} and j^{th} columns of matrix B) as follows:

$$\mu(Pb_i, Pb_j) = \frac{\sum_{n=1}^{m} (Pb_i \wedge Pb_j)_n}{\sum_{n=1}^{m} (Pb_i \vee Pb_j)_n} \tag{22}$$

where the summation runs over n, which is a subscript on the elements of the processes Pb_i and Pb_j and m is the length of each process vector.

It may be noticed that $0 \leq \mu \leq 1$. The value of μ increases when there are more shared system calls between the two processes (due to the numerator) and value of μ decreases when the number of system calls, not shared by both the processes, is more than the shared ones (due to the denominator) in Pb_i and Pb_j.

We also make use of cosine metric, defined in section 3.2.3. For the sake of better representation of the final similarity expression, we denote cosine metric, between the processes P_i and P_j, as follows:

$$\lambda(P_i, P_j) = \frac{P_i \cdot P_j}{\parallel P_i \parallel \cdot \parallel P_j \parallel} \tag{23}$$

4.2.3 Binary Weighted Cosine Metric

We define our new similarity measure, termed as *Binary Weighted Cosine* (BWC) metric, $Sim(P_i, P_j)$ as follows:

$$Sim(P_i, P_j) = \mu(Pb_i, Pb_j) \cdot \lambda(P_i, P_j) \tag{24}$$

The motive behind multiplying μ and λ is that $\lambda(P_i, P_j)$ measures the similarity based on the frequency and $\mu(Pb_i, Pb_j)$ is the weight associated with P_i and P_j. In other words, $\mu(Pb_i, Pb_j)$ tunes the similarity score $\lambda(P_i, P_j)$ according to the number of similar and dissimilar system calls between the two processes. More the number of common system calls, higher will be the value of $\mu(Pb_i, Pb_j)$. Therefore, if a particular system call appears with high frequency, but the number of common system calls between the two processes Pb_i

and Pb_j is low then, $\mu(Pb_i, Pb_j)$ softens the effect of the system call on $\lambda(P_i, P_j)$. Therefore, the similarity measure $Sim(P_i, P_j)$ takes frequency and the number of shared system calls into consideration while calculating similarity between two processes.

In the following section, we analyzes and identify some cases in which the scheme based on kNN-classifier with cosine metric [120] leads to erroneous conclusions. This observation also constitutes a part of our motivation for the present work.

4.3 Analysis of the scheme based on k-NN Classifier With Cosine Metric

An approach based on *kNN classifier* is proposed by Liao and Vemuri [120] where the frequencies of system calls used by a program (process) are used to define program's behavior. The scheme is described in detail in the section 2.3 on page 35. Since the similarity measure, given by equation (2), considers only the frequencies of the system calls appearing in the processes, we observe the following example in which it may produce erroneous results while calculating similarity.

Consider the sequence of system calls associated with the following two processes P_1 and P_2

P_1 = `open close close close close access access access access`

P_2= `open ioctl mmap pipe access login su su audit audit`

Let us consider a third process P given by the sequence of calls as

P= `open close ioctl mmap pipe pipe access access login chmod`

The similarity score of the new process P with P_1 is:

$$CosSim(P, P_1) = 0.6048$$

and the similarity score of the new process P with P_2 is:

$$CosSim(P, P_2) = 0.5714$$

We observe that out of eight, there are only three common system calls between P and P_1 and six common system calls between P and P_2. Intuitively, P_2 is more similar to P than P_1, but the similarity measures, calculated above, indicate the contrary. This is due to the frequent occurrence of `close` and `access` in P_1, and absence of `close` in P_2. The above example makes it clear that while calculating the similarity score, there is no weight accorded to processes having more number of common system calls. We believe that while calculating the similarity score, if we include a factor that depends on the number of common calls, such anomalous results can be avoided. Therefore, in our scheme, we define a similarity measure that depends not only on frequencies of system calls but also on the number of shared system calls between the processes.

With this background and definitions, we describe in the next section the proposed method for anomaly intrusion detection.

4.4 kNN Classifier and BWC metric for Intrusion Detection

As discussed in section 3.3, the matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ are constructed using normal processes and the set S. For every new process P, if P contains a system call that is not in S, the process P is classified as abnormal; if not, it is first converted into a vector for further processing. The binary equivalent Pb of this vector is calculated. Next, the similarity score $\lambda(P, P_j)$ is calculated for every normal vector P_j by using equation (23). If $\lambda(P, P_j) = 1$, P is classified as normal. Otherwise, using equations (22) and (24), the values of $\mu(P, P_j)$ and $Sim(P, P_j)$ are calculated. Values of $Sim(P, P_j)$ are sorted in descending order and the k nearest neighbors (first k highest values) are chosen. We calculate the average value (Avg_Sim) of the k nearest neighbors. The kNN classifier categorizes the new process P as either normal or abnormal according to the rule given below.

`If Avg_Sim` $>$ `Sim_Threshold, classify P as normal, otherwise P is abnormal.`

where Sim_Threshold is a predefined threshold value for similarity score. The pseudo code for the proposed scheme is shown in Figure 14. It should be observed that the algorithm, presented in figure 14, represents a case of *distance-weighted k- nearest neighbor* algorithm, for the *cosine metric* is weighted by another metric. Using the above-defined scheme, if we re-calculate the similarity of process P with the processes P_1 and P_2, given in section 4.3,

```
Given a set of processes and system calls $S$, form the matrices $A = [a_{ij}]$ and $B = [b_{ij}]$
for each process $P$ in the test data do
    if $P$ has some system calls which does not belong to $S$ then
        $P$ is abnormal; exit.
    else then
        for each process $A_j$ in the training data $A$ do
            calculate $Sim(P, A_j)$;
            if$Sim(P, A_j)$ equals 1.0 then
                $P$ is normal; exit.
        end do
        find first $k$ highest values of $Sim(P, A_j)$;
        calculate Avg_Sim for $k$ nearest neighbors so obtained;
        if Avg_Sim is greater than Sim_Threshold then
            $P$ is normal;
        else then
            $P$ is abnormal;
end do
```

Figure 14: Pseudo code of the proposed scheme

using the equations (22), (23) and (24), we get

$$Sim(P, P_1) = 0.2268 \; and \; Sim(P, P_2) = 0.3429$$

The results obtained above validate our hypothesis empirically.

4.5 *Experimental Setup and Results*

I love fools' experiments; I am always making them.

- Charles Darwin (1809-1882, British biologist)

We use BSM audit logs from the 1998 DARPA data [35] for training and testing of our algorithm. The whole experimentation setup remains same as described in section 3.5.1. We perform the experiments with k = 5, 10 and 15. Table 19 shows the results for k = 5 and 10. We have not shown the results for k = 15 as we do not find any significant difference from the case of k = 10. In order to make a valid comparison, we repeat the experiment with Liao and Vemuri's scheme and table 20 summarizes the results for k =10. The results, which we get here, are not complying with the results reported in [13]. The false positive rate is very high at a detection rate of 100%. This may be due to the reason that we may

Threshold	k = 5		k = 10	
	False positive rate	Detection rate	False positive rate	Detection rate
0.52	0.00	0.36	0.00	0.36
0.55	0.00	0.36	0.007	0.38
0.60	0.0003	0.38	0.009	0.76
0.65	0.007	0.52	0.020	0.83
0.70	0.011	0.85	0.024	0.87
0.74	0.022	0.89	0.049	0.94
0.78	0.048	0.96	0.055	0.96
0.80	0.048	0.96	0.10	1.00
0.84	0.052	0.96	0.14	1.00
0.86	0.077	0.96	-	-
0.89	0.087	1.00	-	-

Table 19: False Positive Rate vs Detection Rate for k= 5, 10 for BWC scheme

have chosen a different attack data set from that chosen by Liao et al. This attack data set may contain some processes that show a high similarity with normal processes.

Threshold	False Positive Rate	Detection Rate
0.50	0.00	0.34
0.55	0.0009	0.34
0.70	0.001	0.36
0.78	0.001	0.74
0.85	0.002	0.74
0.88	0.031	0.76
0.90	0.034	0.78
0.93	0.035	0.81
0.95	0.035	0.83
0.97	0.044	0.92
0.98	0.091	0.96
0.99	0.33	0.96
0.992	0.36	0.98
0.994	0.39	1.00

Table 20: False Positive Rate vs Detection Rate for k= 10 for Liao and Vemuri scheme

4.6 Discussion and Analysis

We show the comparative results of BWC scheme with that for Liao and Vemuri and Hu et al. As can be seen in the figure 15, the ROC for k = 5 tends to be lower than that for k = 10, therefore we draw ROC curves for BWC scheme at k = 5 for comparison with Liao and Vemuri method, which is illustrated in figure 16. The ROC curve is a graph between detection rate and false positive rate. For each threshold value we get the detection rate and false positive rate.

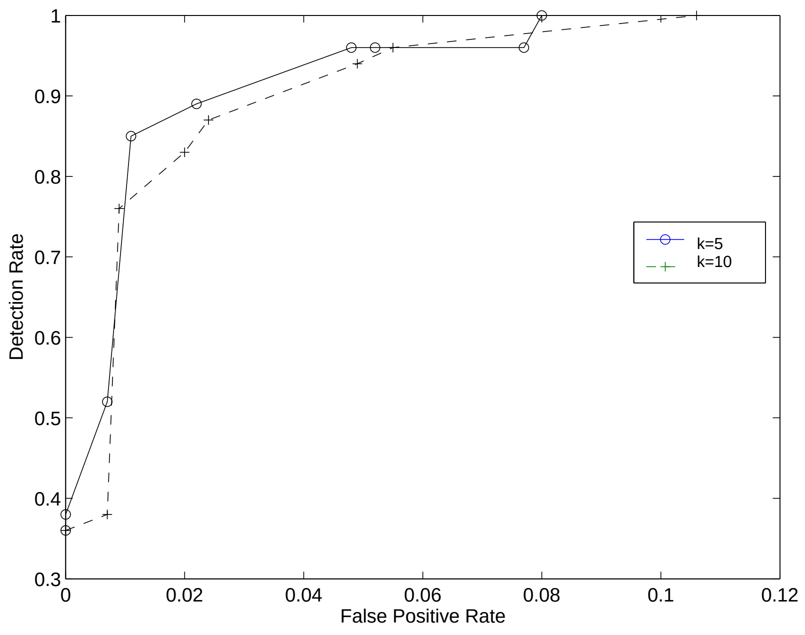

Figure 15: ROC curve for BWC Scheme at k =5, 10

The BWC scheme achieves a detection rate of 100% at a false positive rate of 8% whereas Liao and Vemuri scheme achieves 100% detection rate at 39% false positive. The AUC score for BWC scheme is 0.99, whereas it 0.97 for Liao and Vemuri's scheme. If we look carefully at table 19 for k = 5, we find that from threshold value 0.78 to 0.86, the detection rate is constant. When we investigate, we find that there are two attacks included in the testing data viz. 4.5_it_162228loadmodule and 5.5_it_fdformat_chmod, which could not be detected by BWC method whereas Liao and Vemuri scheme detected them at a lower threshold. On consulting the document on DARPA attack description, which provides various details about the attacks, we find that though the attack 4.5_it_162228loadmodule was launched, it failed to compromise the system. In other words, the processes associated with the attack behaved normally. Therefore it is not a surprise that BWC method could not detect the attack because there indeed was no attack. In case of the second attack 5.5_it_fdformat_chmod, we find that this attack is launched in various stages and

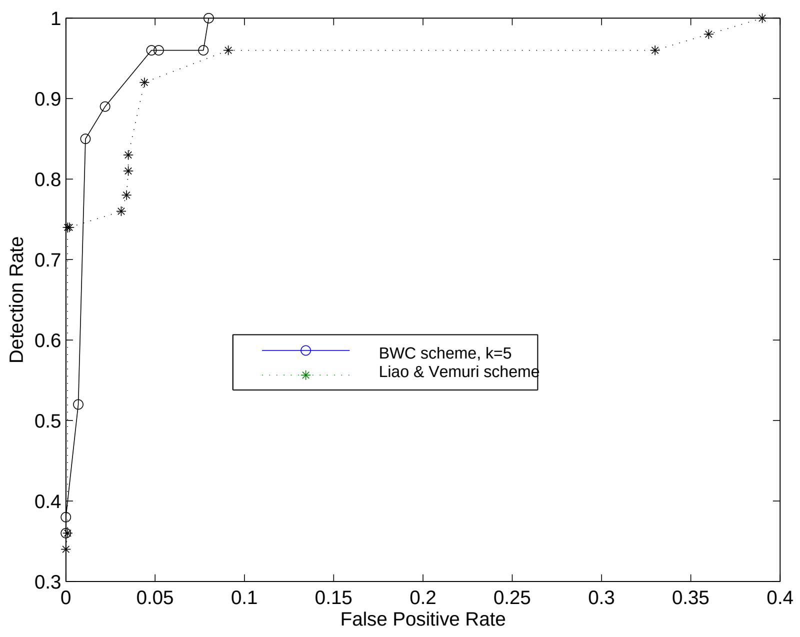

Figure 16: ROC curve for BWC Scheme at k=5 and Liao and Vemuri scheme at k=10

the included instance was in stage 2. Therefore, we may think that attack has not manifested and thus may not be detected. In fact, these events validate that the proposed BWC method checks more closely and rigorously the similarity between the processes. This is attributed to the binary similarity metric as its value increases with the number of system calls shared by the processes being compared. We repeat the whole experiment after removing these two attacks from our testing data set and the results are shown in the Table 5. It can be seen in Table 5 that detection rate reaches 100% with false positive rate as low

Threshold value	k = 5		k = 10	
	False positive rate	Detection rate	False positive rate	Detection rate
0.52	0.00	0.37	0.00	0.37
0.55	0.00	0.37	0.007	0.39
0.60	0.0003	0.39	0.009	0.79
0.65	0.007	0.54	0.020	0.86
0.70	0.011	0.88	0.024	0.90
0.74	0.022	0.92	0.049	0.98
0.78	0.048	1.00	0.055	1.00

Table 21: False Positive Rate vs Detection Rate for k= 5, 10 for BWC scheme after removing two attacks

as 4% only. With this reduced test data set also, Liao and Vemuri scheme gives almost the same results as with the previous data. This could be due to the reason that their scheme detected these attacks at early stage at low threshold with original test data set. Therefore even after removing these attacks, it does not improve in performance.

Another set of experiments is carried out to show the effectiveness of the BWC scheme for noisy data.

4.6.1 Results vis-à-vis RSVM

This part of the section aims to attest the idea that an approach as simple as kNN classifier can be used to produce results that are comparable to the results obtained by more complex method such as the RSVM.

The Robust Support Vector Machine (RSVM) has been applied to anomaly-based IDS. The emphasis is to exhibit the effectiveness of the method in the presence of noise in data. The experiments have been performed on the same BSM DARPA'98 data set that we discussed earlier. A detailed description can be found in [79]. Briefly, we mention below the data sets used for the above experiments and the results thus obtained.

There are two training sets - clean data set and noisy data set. The clean data set contains 300 normal processes and 12 abnormal processes with correct labels. The noisy data set consists of 300 normal processes and 28 abnormal processes. Of these 28 abnormal processes, 16 have been mislabeled as normal (and thus noisy) and the remaining 12 are correctly labeled as abnormal. The testing data set contains 5285 normal processes and 22 intrusive sessions from the two-week testing data of DARPA. We reproduce the SVM and RSVM results [79] in table 22 for ready reference. It has been noticed by Eskin et al [55]

Method	Clean Data		Noisy Data	
	False positive rate	Detection rate	False positive rate	Detection rate
RSVM	3.0%	100%	8.0%	100%
SVM	14.2%	100%	100%	100%

Table 22: False Positive Rate vs Detection Rate of RSVM and SVM under clean and noisy data

that in practical situations the normal data outnumbers the intrusion data by a factor of

100:1. If we look at DARPA data we find that the frequency of intrusive sessions is very low compared to that of normal sessions. In other words, if we collect normal data in a real working environment and if the data is contaminated by the presence of some intrusive data, the proportion of intrusive data will be very small. Whereas the above noisy data set used by Hu is not in this proportion. We also find it difficult to decide the appropriate ratio of normal and abnormal processes in the data. We, therefore, choose a middle path and performed the experiments on a data set in which we try to maintain the ratio of normal and abnormal processes as 100:2 so that proportion of normal and abnormal processes is near to the realistic scenario (namely, 100:1) and we retained the 100:5 proportion for the RSVM experiments. Our training data set consists of 622 normal processes (606 normal + 16 abnormal processes, mislabeled as normal). Out of these 622 processes, 606 are the same as those used in earlier experiments. The testing data set consists of 22 intrusive sessions launched over the two-weeks of testing period in DARPA setup and 5285 normal processes. Table 23 shows the experimental results with the aforementioned data sets. It

Threshold value	$k = 5$		$k = 10$	
	False positive rate	Detection rate	False positive rate	Detection rate
0.52	0.00	0.00	0.00	0.00
0.60	0.0003	0.00	0.007	0.45
0.65	0.006	0.36	0.020	0.50
0.70	0.017	0.59	0.03	0.72
0.74	0.029	0.68	0.05	0.90
0.78	0.054	0.68	0.06	1.00
0.80	0.055	0.72	0.08	1.00
0.84	0.059	0.90	0.14	1.00
0.86	0.083	0.90	-	-
0.89	0.09	1.00	-	-
0.92	0.32	1.00	-	-

Table 23: False Positive Rate vs Detection Rate of BWC scheme under noisy data

can be seen from table 23 that the detection rate with $k = 10$, reaches 100% with a false positive rate of only 6%, whereas in case of RSVM the detection rate is 100% with a false positive rate of 8.0%. It should be noted that in case of RSVM number of training instances are 316 whereas they are 622 in our experiment. We also performed an experiment with 316 training instances (300 normal + 16 abnormal processes as described above) at k = 10 and we could get a detection rate of 100% with a false positive rate of 12%. These figures reveal that BWC scheme needs larger number of normal processes for training as compared to RSVM method.

We also obtained results for the clean training data set (consisting of 606 normal processes and 12 abnormal processes) with k = 5. Each new process is first compared with each of the 12 processes by calculating similarity scores. If the score equals one, the new process is classified as abnormal otherwise regular method described in section 4.4, is followed. Table 24 summarizes the results. With clean data RSVM achieves 100% detection rate with

Threshold	False Positive Rate	Detection Rate
0.60	0.0003	0.18
0.65	0.007	0.72
0.70	0.01	0.90
0.74	0.02	0.95
0.78	0.05	1.00

Table 24: False Positive Rate vs Detection Rate of BWC scheme under clean data

3% false positives, whereas the BWC scheme reaches a detection rate of 100% at 5% false positive rate. Therefore, from performance standpoint, the BWC scheme is slightly inferior to RSVM. Another criterion for performance measurement, mentioned in [79] is that an intrusion detection system should not produce false positive rate greater than 1% with as high detection rate as possible. Table 25 shows the detection rates of BWC, RSVM and SVM methods over the clean data set at false positive rate under 1%. In this scenario, the BWC method performs better than RSVM. From these observations, it can be inferred

Method	False Positive Rate	Detection Rate
BWC	90.0%	1%
RSVM	81.8%	1%
SVM	81.8%	1%

Table 25: Detection rate of BWC, RSVM and SVM at a fixed false positive rate of 1%

that our proposed scheme is able to perform well in the presence of noisy data, but the amount of training data required is more than in the case of RSVM method. If we look at the complexity, our proposed method is simpler than the RSVM method.

In the view of above experiments and the analysis, it can be asserted that the proposed similarity metric performs better than simply applying cosine similarity metric. It can also be noticed that minimum threshold value at which the proposed metric could identify all the abnormal processes (0.78) is lower than that in case of Liao and Vemuri scheme (0.994). It shows that separation boundary created by proposed similarity metric is much wider, thus facilitating to minimize the overlap between normal and abnormal processes.

4.6.2 Cross-Validation Experiments

For this scheme also, we perform the cross-validation approach to estimate the generalization error. The experimentation setup is same as described in the section 3.5.1.1. The results are summarized in the table 26. Each entry is the aggregation of the outcome of 10 rounds of experiment. If we compare tables 19 and 26, we find that for $k = 10$, the BWC

K=5		K=10	
FP(%)	DR(%)	FP(%)	DR(%)
0.0	46.12	**0.0**	45.76
1.0	46.9	**1.0**	52.0
7.82	**90.0**	8.58	**90.0**
18.88	**100.0**	18.13	**100.0**

Table 26: False Positive Rate vs Detection Rate for different combinations under the leave-one-out experiment

scheme is able to yeild almost same *accuracy* (100% DR at 14% FP and 100% DR at 18.13% FP) under *leave-one-out* experimentaions, which shows that the scheme is able to classify unseen cases, resulting in a less generalization error.

4.7 Including Ordering of System Calls by Considering Relative Positions of System Calls

In the aforementioned approach, we considered frequencies of system calls. However, it may also be of interest to observe the temporal locations of system calls. The execution path of a process should almost be same under various normal conditions. Any deviation in this path indicates the possibility of a malicious activity. It is obvious that any deviation will disturb the ordering of system calls in local temporal regions i.e. the relative positions of system calls get affected. This change in ordering is likely to get reflected in malicious process. Therefore, the order of system calls occurring in a process is very important while classifying a process as normal or abnormal. We make use of the *Kendall Tau Distance* [51] to determine similarity in terms of ordering of system calls locally.

4.7.1 Kendall Tau Distance

In many situations, it is desired to rank the various alternatives according to some criterion. If there is only one *judge* to rank them, the task is easy- simply accept the *ranked list*, decided by the judge. But if there are many people to decide or to vote on the available alternatives, the task to choose the one *optimal* ranked list of alternatives is no more a trivial task. The problem of computing a *consensus* ranking of the alternatives, given the individual ranking preferences of several judges, is called the *rank aggregation problem* [51]. In order to obtain a *consensus* ranking, it is desirable to compare two rank lists by considering similarity (or dissimilarity) between them. *Kendall Tau Distance* is one such method to calculate the similarity (or dissimilarity) among ranked lists. It is calculated by counting the number of pairwise disaggrements between two lists. In this context, a process can be thought of as a ranked list of different system calls, in which any subsequent appearence of a system call is discarded. For example, let P be a process given as:

P=`open close close ioctl mmap mmap pipe pipe access access open open login chmod close`

then the ranked list, corresponding to P is given by

P'=`open close ioctl mmap pipe access login chmod`

In the following paragraph, we give a more formal formulation of the problem.

Given a set S of system calls, a process (ordered set of system calls) t with respect to S is an ordering of a subset M of S, i.e. $t = [x_1 \geq x_2 \geq \ldots \geq x_d]$, with each system call $x_i \in M$, and $x - i \geq x_j$ implies that in the process t, system call x_i appears before the system call x_j. If $i \in S$ is present in t, then $l(i)$ denotes the position of i. The Kendall Tau Distance counts the number of pair-wise disagreements between two processes t and m as follows:

$$K(t,m) = \frac{|\{i,j\} : i < j,\ t(i) < t(j),\ m(i) > m(j)\ or\ i > j,\ t(i) > t(j),\ m(i) < m(j)|}{\binom{|M|}{2}} \quad (25)$$

In order to calculate the Kendall Tau distance between two processes P_i and P_j, first of all P_i is converted into P'_i which contains the first occurrence of each system call from P_i. It can be observed that in practice, it is difficult to find the set M such that each process has system calls only from the set M. We, therefore, modify equation (25) to calculate Kendall distance as follows:

$$K(P_i, P_j) = \frac{|\{r,s\} : Condition(W)\ or\ Condition(W')|}{|P'_i||P'_i|} \quad (26)$$

where $Condition(W) = r < s, P'_i(r) < P'_i(s), P'_j(r) > P'_j(s)$ and $Condition(W') = r > s, P'_i(r) > P'_i(s), P'_j(r) < P'_j(s)$. All the (r, s) pairs, for comparison, are obtained from the set of system calls S.

With this new distance in hand, we modify our kNN and BWC metric based scheme, described in section 4.4, as shown in figure 17.

4.7.2 Experimental Results with Kendall Tau Distance

The experimental setup remains same as used in the previous experiments (section 4.5). We choose $k = 5,\ 10$ for kNN classifier. It turns out that the scheme gives slightly better results for both the values of k. The results are summarized in table 27 for $k = 5$. The first

Threshold	False Positive	Detection Rate
0.20, 0.25	0.000	0.80
0.25, 0.30	0.011	0.96
0.25, 0.35	0.012	0.98
0.60, 0.25	0.019	0.98
0.65, 0.25	0.023	1.00

Table 27: FP vs. DR for BWC metric with Kendall Tau distance, at $k = 5$.

entry in the first column of table 27 denotes the threshold value for *BWC* metric whereas second entry stands for threshold for *Kendall Tau* distance. If we look at the algorithm, presented in figure 17, we observe that both the thresholds work independently to yield the results given in table 27. It is curious to know the combined effect of both the thresholds on the resultant accuracy. We, therefore, modify the the algorithm a litle as follows.

```
Given a set of processes and system calls $S$, form the matrices $A = [a_{ij}]$ and $B = [b_{ij}]$
for each process $P$ in the test data do
    if $P$ has some system calls which does not belong to $S$ then
        $P$ is abnormal; exit.
    else then
        for each process $A_j$ in the training data $A$ do
            calculate $Sim(P, A_j)$;
            if$Sim(P, A_j)$ equals 1.0 then
                $P$ is normal; exit.
        end do
        find first $k$ highest values of $Sim(P, A_j)$;
        calculate Avg_Sim for $k$ nearest neighbors so obtained;
        for each $k$ nearest neighbor, calculate Kendall Distance;
        calculate Avg_Dist for $k$ nearest neighbors;
        if Avg_Sim is greater than Sim_Threshold and
        Avg_Dist is less than Dist_Threshold then
            $P$ is normal;
        else then
            $P$ is abnormal;
end do
```

Figure 17: Pseudo code of the scheme,based on BWC metric and Kendall Tau Distance

After calculating the *Kendall Tau* distance $K(P_i, P_j)$, we calculate its complement as $K_{comp} = 1 - K(P_i, P_j)$ and multiply this quantity with BWC metric $Sim(P_i, P_j)$, given by equation (24). Thus the final similarity measure is given by the following equation:

$$Sim_{Ken}(P_i, P_j) = K_{comp}(P_i, P_j) \cdot Sim(P_i, P_j) \qquad (27)$$

It should be noted that with the equation (27), taken as the similarity measure,the resulting classification algorithm again represents the case of *distance-weighted k- nearest neighbor* algorithm, discussed earlier. Thus we have a single similarity score and we apply different threshold values to get different pairs of FP and DR. The results with these modifications are shown in the table 28.

If we compare results from tables 20 and 28, we find both the schemes are producing same level of accuracy in terms of detection rate. The false positive rate could not reach to 0 without making detection rate also nearly 0, which implies that few abnormal and normal processes are very much similar. The AUC score with the modified Kandell Tau distance scheme is 0.62, which indicates that some of the positive examples are ranked very low,

Threshold	False Positives	Detection Rate
0.01	0.37	0.34
0.05	0.37	0.34
0.10	0.37	0.34
0.20	0.37	0.34
0.25	0.37	0.34
0.35	0.37	0.34
0.45	0.37	0.36
0.50	0.37	0.38
0.55	0.37	0.83
0.65	0.38	0.92
0.70	0.38	1.00

Table 28: FP vs DR with BWC and Kendall Tau distance combined in one similarity score, defined by equation (27).

whereas some of the negative examples are ranked very high (see equation (21)). Also the scheme defined by equation (27) is very expensive in terms of computation. However, if we look at table 27, we find that algorithm presented in figure 17 yielding better results than original kNN-based scheme. Thus, if we combine BWC metric and Kendall Tau distance in one measure, result degrades in terms of accuracy. From this observation, it can be inferred that Kendall Tau distance is not efficient enough to distinguish between normal and abnormal processes.

4.8 Conclusions

All anomaly-based intrusion detection systems work on the assumption that normal activities differ from the abnormal activities (intrusions) substantially. In case of IDS models that learn a program's behavior, these differences may manifest in the form of the frequency of system calls, the number of shared system calls used by the processes and the ordering of system calls used by the processes under normal and abnormal execution. Our BWC scheme considers the first two of these factors while classifying a new process as normal or abnormal. The new BWC metric has two factors. One factor calculates the similarity between the two processes based on the frequency of system calls by making use of the cosine metric. The second factor operates on the binary form of the process vectors and calculates the similarity between the two processes based on the number of the shared system calls in those processes.

With the BWC metric, we obtain a detection rate of 100% at a false positive rate of 4%, which is a significant improvement over the performance of the scheme proposed by Liao and Vemuri, which makes use of kNN classifier with cosine metric. At a fixed false positive rate of 1%, WBC scheme outperforms with a detection rate of 90% as compared to the detection rate of 81.8% with other schemes. We also combine the BWC metric with Kendall Tau distance to capture the ordering of system calls in a process. The results are comparable with kNN based scheme. However, we notice that calculating the Kendall Tau distance is computationally very expensive. Infact, only BWC metric alone is showing good results. The experiments show that the proposed BWC scheme performs consistently even in the presence of noise in the training data and the results, thus obtained, are comparable to one of the recently published study with RSVM. We believe that the proposed method takes a very simple approach by using kNN classifier and is easier to understand and implement than the RSVM. We notice that though the new metric performs very well by capturing frequency and commonality of system calls, a very important aspect of similarity measurement is missing, which is the ordering of system calls. Kendall Tau distance tries to capture the partial ordering information about the system calls, though not very efficiently. In the next chapter, we try to explore technique that captures the ordering of system calls in the process.

The described work in the chapter is published as:
Rawat S, Gulati V P, Pujari A K. (2004), ”Frequency And Ordering Based Similarity Measure For Host Based Intrusion Detection”. Journal of Information Management and Computer Security, 12(5). Emerald Press. 2004. pp. 411-421
Rawat S, Gulati V P, Pujari A K, Vemuri V Rao (2004) ”Intrusion Detection using Text Processing Techniques with a Binary-Weighted Cosine Metric” Submitted to: International Journal of Information Security, Springer.

CHAPTER V

INTRUSION DETECTION USING ROUGH SET THEORY

Keep on the lookout for novel ideas that others have used successfully. Your idea has to be original only in its adaptation to the problem you're working on.

- Thomas Edison (1847-1931)

5.1 *Introduction*

In this chapter, we discuss the usability of rough set theory to intrusion detection. Rough set theory is being used in classification task in other domains and is capable to classifying data in the situations where there is ambiguity in the concept, represented by the data.

Forrest *et al* [59][60] suggest that system calls trace of a process under normal execution can be taken as its normal behavior in terms of system calls, as variation in sequences of system calls is very small. On the other hand, this variation is relatively higher when compared to a sequence of system calls under abnormal execution. This variation can be attributed to the presence of one or more *alien* (thus malicious) subsequences in the abnormal process. We, therefore, advocate here that the intrusive behavior of a process is localized characteristics of the process. There are certain smaller episodes in a process that make the process intrusive in an otherwise normal stream. As a result it may be unnecessary to consider the whole process in totality and to attempt to characterize its abnormal features. In the present work we establish that subsequences of reasonably small length of system calls would suffice to identify abnormality in a process. Thus intrusive part should be detectable as a subsequence of the whole abnormal sequence of the process.

In this chapter, we present a technique of discovering rules for intrusion detection. We make use of *rough set theory* for this purpose. To best of our knowledge, Lin was the first to propose the idea of applying rough sets to the problem of anomaly detection [124]. Though the paper lacks the experimental results [124], it provides some solid theoretical background. The following two theorems are important:

I. Every sequence of records in computer has a repeating sequence

II. If the audit trail is long enough, then there are repeating records

Following the argument of Forrest *et al* and in the view of above theorems, our approach is based on subsequences of system calls. We formulate the problem as a classification problem by writing the set of subsequences as a decision table. The proposed method is a combination of signature-based and anomaly based approaches. A program behavior is monitored as a sequence of system calls. These sequences are further converted into the subsequences of shorter length. These subsequences are considered as the signatures for malicious as well as normal activities. By doing so, one of the disadvantages of signature-based approach of frequently updating the signature database can be avoided. Empirical results show that the proposed system is able to detect new abnormal activities without updating the signatures. Further, these signatures are represented in the form of IF-THEN type decision rules. The advantage of representing signatures in this form is that such signatures are easy to interpret for further analysis. Rough set theory is used to induce decision rules. Rules induced by using rough set theory are very compact because before inducing rules, all the redundant features of the audit data are removed. This makes the matching of rules *faster*, thus making the system suitable for on-line detection. The proposed system is also fast in the sense that process is compared, in parts, as it starts calling system calls. So we do not have to wait until it exits.

Rest of the chapter is organized as follows: Section 5.2 presents some preliminary background to understand the approach, which includes rough set theory, decision table and rules learning algorithm LEM2. A relationship between IDS and *information system* is established in section 5.3. The process of rule formation is detailed in section 5.4. A detailed description of the proposed scheme is given in the section 5.5. Section 5.6 covers the experimental results and analysis of the results. Section 5.7 concludes the chapter.

5.2 *Preliminary Background*

In this section, we provide some basic definitions and notations used in our work. Later in the section we try to formulate intrusion detection as a problem in rough set theory.

5.2.1 Rough Set Theory

Knowledge discovery comprises of techniques from machine learning, statistics, pattern recognition, fuzzy and rough sets etc to extract knowledge or information from the huge amount of data. Rough set theory was introduced by Z. Pawlak [151] to provide a systematic mathematical framework for studying imprecise and insufficient knowledge to generate decision rules. A rough set is a set of objects that cannot be precisely characterized based on a set of available attributes. The idea of rough set consists of the approximation of a set by a pair of sets, called as *lower approximation* and *upper approximation*.

Let $S =< U, Q, V, f >$ be an information system where U is the closed universe, a finite set of N objects $x_1, x_2, ..., x_N$; Q is a finite set of n attributes $q_1, q_2, ..., q_n$; $V = \bigcup_{q \in Q} V_q$ where V_q the domain of attribute q; and $f : U \times Q \rightarrow V$ is the total function called as information function such that $f(x, q) \in V_q$ for every $x \in U$ and $q \in Q$. A subset of attributes $A \subseteq Q$ determines as equivalence relation of the universe U, called as *indiscernibility relation* and denoted as *IND(A)*.

Definition 1: For a given $A \subseteq Q$ and $X \subseteq U$ (a concept X), the A-lower approximation ($\underline{A}X$) and A-upper approximation ($\overline{A}X$) of set X are defined as follows:

$$\begin{aligned} \underline{A}X &= \{x \in U : [x]_A \subseteq X\} = \bigcup\{Y \in A^* : Y \subseteq X\} \\ \overline{A}X &= \{x \in U : [x]_A \cap X \neq \phi\} = \bigcup\{Y \in A^* : Y \cap X \neq \phi\} \end{aligned} \tag{28}$$

where $[x]_A = \{y \in U : xIND(A)y\}$ and A^* is the partition of U generated by *IND(A)* on U.

Definition 2: The accuracy of an approximation of the set X by the set of attributes A is defined as:

$$\alpha_A(X) = \frac{|\underline{A}X|}{|\overline{A}X|} \tag{29}$$

Definition 3: Given an information system $S =< U, Q, V, f >$, with condition and decision attributes $Q = A \cup D$, s.t. $A \cap D = \phi$, for a given set of condition attributes A, we define

A-positive region $POS_A(D)$ in the relation $IND(D)$ as:

$$POS_A(D) = \bigcup\{\underline{A}X : X \in D^*\} \tag{30}$$

where D^* denotes the family of equivalence classes defined by the relation *IND(D)*. $POS_A(D)$ contains all the objects in U which can be classified perfectly without error into the distinct classes defined by *IND(D)*, based only on information in relation *IND(D)*. Similarly, in general, if $A, B \subseteq Q$, then *A-positive region* of B is defined as

$$POS_A(B) = \bigcup_{X \in B^*} \underline{A}X \tag{31}$$

5.2.2 Decision Table

A data set is represented as a table, where each row represents an object or case. Every column represents an attribute that can be measured for each object. In supervised learning, there is an outcome of classification that is known. This a posteriori knowledge is expressed by one distinguished attribute called decision attribute. A table wherein one of the attributes is decision attribute is called a *decision table*. More precisely we can define it as follows:

Definition 4: Given an information system $S =< U, Q, V, f >$, if Q can be expressed as condition and decision attributes i.e. $Q = A \cup D$, with $A \cap D = \phi$, then S is called a decision table (or decision system) [31].

Definition 5: A decision table is said to be *consistent* if each unique row has only one value of decision attribute. Objects from decision table can be partitioned into disjoint classes, called *concepts*, based on the decision attributes D.

Definition 6: An expression $(a = v)$, where $a \in A$ and $v \in V_q$, is called an *atomic formula* (or elementary condition) c. An elementary condition c can be interpreted as mapping:

$$c : U \rightarrow \{true, false\}$$

A conjunction C of q elementary condition is denoted by

$$C = c_1 \wedge c_2 \wedge ... \wedge c_q.$$

Definition 7: The cover of C, denoted by $[C]$, is the subset of objects (examples), which satisfy the conditions represented by C.

$$[C] = \{x \in U : C(x) = true\}$$

5.2.3 Decision Rules

The decision rules are logically described as

If (a conjunction of elementary conditions) **then** (decision)

Decision rules can be considered as data patterns, which represent relationships between values of attributes in the decision table. The rule set, obtained from a consistent table, is said to be *deterministic*. Any set of rules may fall into any of the following three categories [180]:

- Minimum set of decision rules,
- Exhaustive set of decision rules,
- Satisfactory set of decision rules

First category contains the smallest number of rules sufficient to cover the set of objects belonging to one class. Second category consists of all the rules that can be induced from the table. However, time complexity for the second choice is exponential and using this approach may not be practical for larger data set [180]. For our experiment we, therefore, choose the first approach of inducing rules, as the data set used in our experiment is very large. We make use of a rough set based algorithm LEM2 for inducing rules [72], which is presented in figure 18. We use the same notations introduced in the previous section. Additionally, $C(G)$ denote the set of conditions c currently considered to be added to the conjunction C. K is the concept and rule r is characterized by its conditional part R. LEM2 algorithm follows a heuristic strategy from machine learning techniques. The strategy starts with creating a first rule by choosing sequentially the 'best' elementary conditions (conjunction of attributes values). Then, all the learning examples that matches this rules are removed from consideration. The process is repeated iteratively while some

```
**Input**:$K$- set of objects
**Output**:$R$- set of rules
begin
   $G = K$;
   $R = \phi$;
   while $G \neq \phi$ do
      begin
      $C = \phi$;
      $C(G) = \{c : [c] \cap G \neq \phi\}$;
      while $(C = \phi)$ or $(!([C] \subseteq K))$ do
         begin
         select a pair $c \in C(G)$ such that $|[c] \cap G|$ is maximum;
         if ties, select a pair $c \in C(G)$ with the smallest cardinality $|[c]|$;
         if further ties occur, select the first pair from the list;
         $C = C \cup \{c\}; G = [c] \cap G$;
         $C(G) = \{c : [c] \cap G \neq \phi\}$;
         $C(G) = C(G) - C$;
         end;
         for each elementary condition $c \in C$ do
            if $[C - c] \subseteq K$ then $C = C - \{c\}$;
         create rule $r$ basing the conjunction $C$ and add it to $R$;
         $G = K - \bigcup_{r \in R} [R]$;
      end;
      for each $r \in R$ do
      if $\bigcup_{s \in R - r} [S] = K$ then $R = R - r$
   end
```

Figure 18: LEM2 algorithm

learning examples remain uncovered. The rules so obtained are capable of classifying new unseen objects [180]. In the whole process, the algorithm also discards all the dispensable attributes. Thus the number of attributes to be matched is reduced which makes the algorithm faster and hence more suitable for nearly real time detection.

5.3 *Representing IDS as Information System*

The data mining techniques are well suited for IDS design because the aim of an intrusion detection system is to trigger alarm (present knowledge) when any intrusion occurs. Thus an IDS can be thought of a decision support system which stores huge data (host or network related) and extracts useful patterns (information about the normal and abnormal behavior) so that it can classify normal and abnormal data precisely. Forrest *et al* [60] suggested the use of small sequences of system calls, made by a process, as the profile of the process. The study done by Lee *et al* [117] also validates this observation. But if we analyze normal and abnormal processes, we find that not all parts of an abnormal process are responsible for intrusion. Thus intrusive part should be detectable as a subsequence of the whole abnormal process. Thus one point of focus of this study is to determine the adequate length of such subsequences. Also as pointed out earlier, not all of the subsequences of an abnormal process are abnormal. Many of them will be identical to those occurred in normal process. This is the point where rough set theory can be used to derive disjoint set of subsequences.

Let P be a set of normal processes, defined as sequences of system calls. Then the l^{th} process can be represented as

$$< p_1^l, p_2^l, \ldots, p_n^l >$$

where n is the length of the process and p_j^l is j^{th} system call. Each of these processes is transformed into the subsequences of length k. Thus for l^{th} sequence, i^{th} subsequence is given by

$$p_i^l, p_{i+1}^l \ldots, p_{i+k-1}^l$$

Each of these subsequences is labeled as normal. In case of abnormal processes, many of the subsequences are identical to those occurred in normal process. Thus a subsequence corresponding to an abnormal process, matching with any of the normal subsequences, is discarded; otherwise it is labeled as abnormal. With the above formulation, we consider an intrusion detection system as an information system $S =< U, Q, V, f >$, defined in the section 5.2.1 with $Q = A \cup D$, D= {normal, abnormal}. The number of attributes in the conditional part A equals the length of the subsequence. V consists of all the system calls appearing in all the processes used for training. U consists of all the subsequences of the chosen length that can be derived from all the processes using sliding window of the chosen length k. It should be noted that by removing duplicate subsequences, we get a consistent decision table because no subsequence can belong to normal as well as abnormal classes. To put in rough set terminology, let P and T be two normal and abnormal processes respectively given as $P =< p_1, p_2, \ldots, p_n >$ and $T =< t_1, t_2, \ldots, t_m >$ where p_i's and t_j's are system calls. Let k be the size of sliding window. Each process P and T is transformed into subsequences of length k. Then the information system S can be represented as follows:

A is comprised of k attributes $A_1, A_2 \ldots, A_k$. U consists of all the subsequences of the forms $P_i = p_i, p_{i+1}, \ldots, p_{i+k-1}$ and $T_i = t_i, t_{i+1}, \ldots, t_{i+k-1}$, represented as rows of S. Let us denote normal and abnormal classes by D_{normal} and $D_{abnormal}$, which is a partition of U by the decision attribute D, denoted as D^*. As mentioned earlier, there will be many T_i's which are identical to some P_i's, but labeled as abnormal. Therefore the lower approximations of both the classes D_{normal} and $D_{abnormal}$ are calculated as $\underline{A}D_{normal}$ and $\underline{A}D_{abnormal}$. Therefore the positive region

$$POS_A(D) = \underline{A}D_{normal} \cup \underline{A}D_{abnormal}$$

contains only those subsequences that belong to either of the classes but not both. At this stage, we apply a rule learning algorithm to extract rules to detect abnormal behavior.

5.4 Learning IF-THEN Rules for Intrusion Detection

We apply LEM2 algorithm on S to form the certain rules. By doing so, we get the signature for normal and abnormal processes. Let us take an example to make it more clear. Let
P^1 = <`fcntl, close, close, fcntl, close, fcntl, close, open`> and
P^2 = <`fcntl, close, fcntl, close, open, open`>
be normal and abnormal processes respectively. We transform P^1 into a set of subsequences using a sliding window of length 5. We label all the 4 subsequences as normal. While calculating the subsequences of P^2, the first subsequence < `fcntl, close, fcntl, close, open`> matches with the last subsequence of P^1 and therefore it is discarded. The second subsequence < `close, fcntl, close, open, open`> is labeled as abnormal and added to the decision table. The final decision table is shown in the table 29. It can be seen in table 29 that the decision table, thus created, is consistent. We calculate IF-THEN rules, using LEM2 algorithm, shown in table 30. LEM2 algorithm, in its original form, does not yield any *default rule*. It means that outcome of LEM2 algorithm is the set

Objects	A_1	A_2	A_3	A_4	A_5	Decision
1	fcntl	close	close	fcntl	close	normal
2	close	close	fcntl	close	fcntl	normal
3	close	fcntl	close	fcntl	close	normal
4	fcntl	close	fcntl	close	open	normal
5	close	fcntl	close	open	open	abnormal

Table 29: Representation of subsequences

1	(A_2 = close)$\Rightarrow$ (Dec =normal)
2	(A_1 = close)$\wedge$ (A_2 = fcntl) $\wedge$ (A_3 = close) $\wedge$ (A_4 = fcntl)$\Rightarrow$ (Dec = normal)
3	(A_1 = close) $\wedge$ (A_2 = fcntl) $\wedge$ (A_3 = close) $\wedge$ (A_4 = open) $\Rightarrow$ (Dec=abnormal)

Table 30: Representation of IF-THEN rules

of exact rules representing the classes in the data. Any instance, which is not covered by any of the rules, will be treated as unclassified. In IDS terminology, such types of rule lead to a misuse-based intrusion detection. Therefore, to force the LEM2 to produce rules such that each future instance will be covered by at least one rule, we introduce a *default rule* in the set of rules. This default rule states that

'any subsequence, which is not covered by any of the rule is abnormal'

With the inclusion of the default rule, the final rule set represents the case of a hybrid IDS because it has rules for normal and abnormal processes i.e. misuse-based system and rule for anything which deviates from learnt behavior i.e anomaly-based system. We show the experimental results for both of the cases in the section on experimental results.

5.5 *Process Classification based on Learnt Rules*

The rules, induced by LEM2, are used to classify new processes. While classifying any new process, it is first transformed into a set of subsequences and each of these subsequences is classified on the basis on decision rules. If any subsequence pertaining to a process is classified as abnormal, the whole process is considered as abnormal. The algorithmic form of the proposed scheme is presented in figure 25 below.

```
Training Phase
  I.  Collect normal and abnormal processes
 II.  Calculate the subsequences of these processes of length $k$ (number of attributes)
III.  Remove duplicates
 IV.  Construct the decision table with labels normal and abnormal
  V.  Calculate decision rule set $D$R using LEM2
Testing Phase
  I.  For each process $P$ in the testing data do
 II.      Convert the process into decision table of length $k$
III.      Classify each subsequence using the rule set $DR$
 IV.      if any of the subsequence is classified as abnormal then
  V.          $P$ is abnormal
 VI.      else $P$ is normal
VII.  end do
```

Figure 19: Algorithmic representation of the proposed scheme

5.6 *Experimental Setup and Results*

The scheme described in the previous section is tested upon the well-cited DARPA'98 data [35]. Process level information can be derived from BSM audit data. A detailed general procedure for the extraction of process from the audit logs is described in section 3.5.1. There are around 2000 normal sessions reported in the four days of data. We extract the processes occurring during these days and our training data set consists of 606 unique processes. There are 412 normal sessions on the fifth day and we extract 339 *unique* normal processes from these sessions. We use these 339 normal processes for the testing data. A total of 28 abnormal processes are extracted from the whole seven-week of data. Out of these 28 processes, 12 are used for training and remaining 16 are used for testing the detection rate of the approach. In case of a normal process, all of its subsequences should be normal. Therefore in order to test the false positive rate, we take normal subsequences of all the 339 processes together and define *coefficient of normal accuracy* as

$$\eta_n = \frac{N_c}{N} \tag{32}$$

Where N_c is the total number of normal subsequences correctly classified as normal and N is total normal subsequences used in testing data. In case of abnormal process, as mentioned earlier, not all the subsequences are abnormal. Therefore, we say that an abnormal process is detected if any of its subsequences is labeled as abnormal and we define *coefficient of abnormal accuracy* as

$$\eta_a = \frac{A_c}{A} \tag{33}$$

where A_c is total number of processes classified as abnormal and A is total abnormal processes used in testing data. It should be noted that coefficient of normal accuracy η_n is inversely proportional to the false positive rate i.e. higher the value of η_n, lower is the false positive rate.

All the above experiments are conducted using RSES and DIXER tools, developed at the University of Warsaw [158].

5.6.1 Experiments without a Default Rule

As discussed in section 5.4, we classify processes solely on the basis of rule set generated by LEM2 algorithm without including a default rule. We perform the experiments for different values of the length of the subsequences. For each value, a set of decision rule is calculated. Using this set of rules, the values of η_n and η_a are calculated. Table 31 shows the results of experiments for different values of length of subsequences. It should be pointed

Length of the subsequence	Number of the subsequences before/after removing duplicates	Number of rules	Value of η_n	Value of η_a	$\frac{(\eta_n+\eta_a)}{2}$
5	170976/2461	929	0.999	0.750	0.8745
10	167886/5968	1702	0.997	0.750	0.8735
15	164801/8441	1828	0.999	0.750	0.8745
20	161724/10525	1797	0.998	0.812	0.9050
25	158669/12299	1707	1.000	0.750	0.8750
30	155640/13939	1789	1.000	0.860	0.9300
35	152625/15398	1810	0.999	0.930	0.9645
40	149628/16641	1762	0.998	0.860	0.9290
45	146654/17666	1699	0.998	0.928	0.9630

Table 31: Values of coefficients of normal and abnormal accuracy for different values of subsequence length without a default rule

out that while doing the above experiment, any subsequence, not covered by any of the rules is assigned to a special class *undefined* and such subsequences are excluded while calculating η_n. The minimum value of η_n is 0.997 i.e. in worst case the false positive rate is 0.003, which implies that per day, there are $339 \times 0.003 = 1.017$ false alarms. This type of approach is well suitable in situation where there is a second level of check to further analyze the data when we are not certain about the event and tolerance level is high in terms of attacks. But such an approach may delay in decision-making due to lack of high confidence.

5.6.2 Experiments with a Default Rule

In order to speed-up the decision-making about the *undefined* processes, we repeat our series of experiments with a default rule '*any subsequence, which is not covered by any of the rule is abnormal*'. This approach is well suited in situation wherein tolerance limit for attacks is very low. Thus no event is classified based on further analysis, but based on the rules including the default one i.e. each event is classified on-line. It may also be noted that while the former approach is a misuse-based, second is a hybrid (anomaly and misuse) approach. Table 32 lists the results of the experiment. The minimum value of η_n is 0.958 i.e. in worst case the false positive rate is 0.042, which implies that per day, there are $339 \times 0.042 = 14.23$ false alarms. If we look at figure 20, we find that for first

Length of the subsequence	Number of rules	Value of η_n	Value of η_a	$\frac{(\eta_n+\eta_a)}{2}$
5	929	0.966	0.875	0.9205
10	1702	0.959	0.812	0.8855
15	1828	0.954	0.812	0.8830
20	1797	0.958	0.875	0.9165
25	1707	0.960	0.875	0.9175
30	1789	0.958	0.930	0.9440
35	1810	0.959	0.930	0.9445
40	1762	0.958	1.000	0.9790
45	1699	0.959	0.928	0.9435

Table 32: Values of coefficients of normal and abnormal accuracy for different values of subsequence length with a default rule

set of experiment, the value of η_n is high (i.e. low rate of false positives (first longest bar

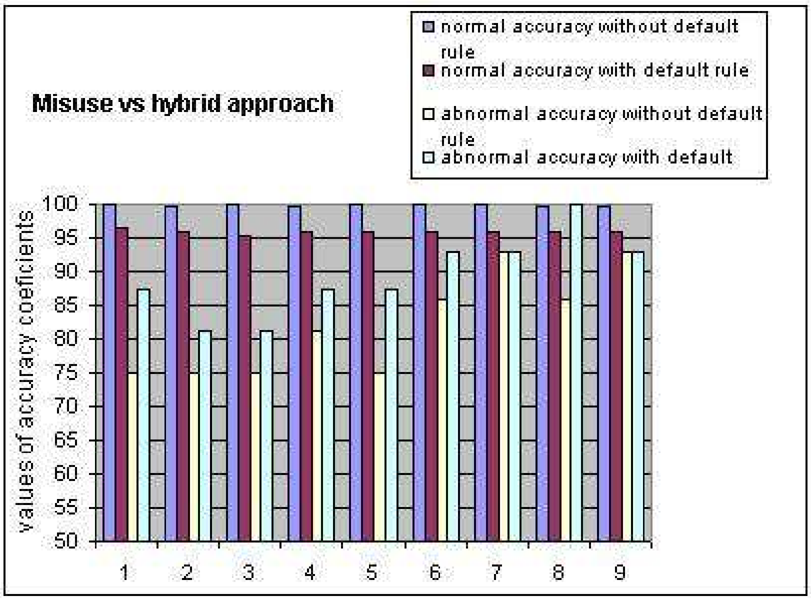

Figure 20: Variation between values of coefficients of normal and abnormal accuracy, with and without a default rule

in the figure 20)) but detection rate (η_a) could not reach 1.0. There is a clear distinction between the values of η_n with and without a default rule. This can be understood as we train our system only with 12 abnormal processes, which is much smaller than 606 normal processes used in training. In the second experiment when we included a default rule, we could get detection rate of 1.0 with normal accuracy as high as 0.958. The presence of the default rule has made the system anomaly-based and from this point of view, a decline in the value of η_n (rise in the rate of false positive) is anticipated as anomaly-based systems are known to have a high rate of false positive. Another inference that can be made from figure 20 is that normal processes follow patterns of system calls more strictly than abnormal processes. Very few normal processes are similar to abnormal processes, but vise-versa is not true. An efficient IDS should have high values of η_n and η_a i.e. low false positive rate and high detection rate. The last column of table 3 and table 4 shows the average taken over the values of η_n and η_a. Figure 21 shows that the results obtained with a default rule (hybrid approach) outperform those obtained without a default rule (but not with a significant difference). The bold lines in the figure 4 represent the trends

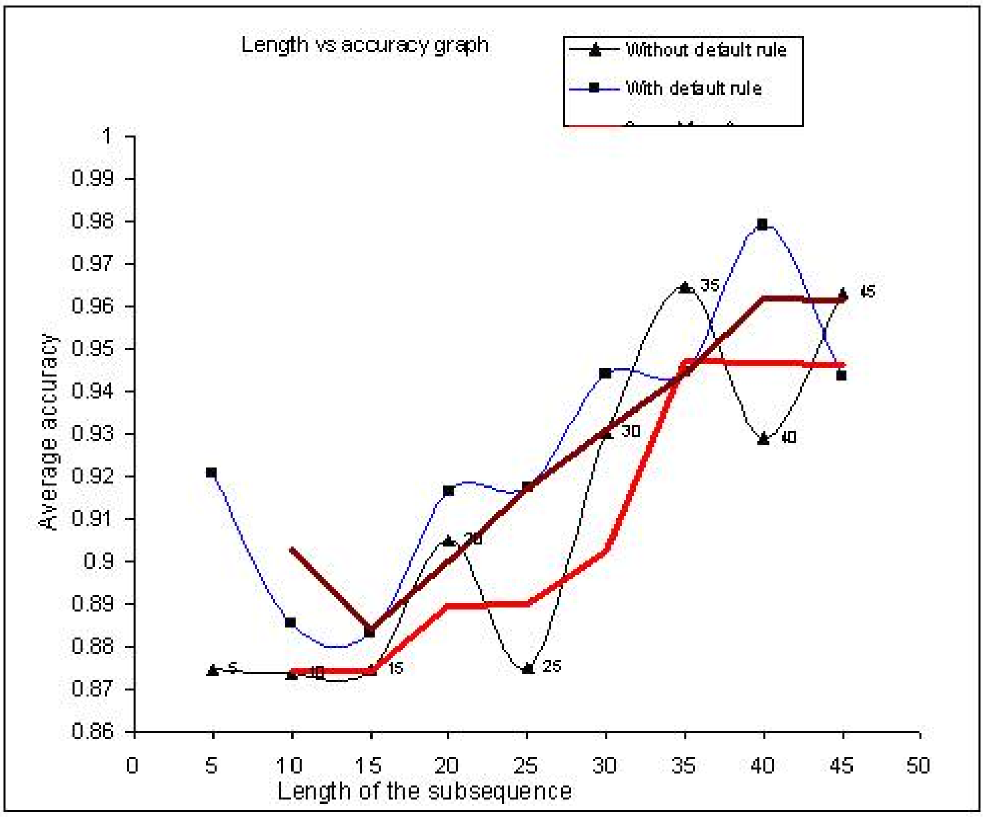

Figure 21: Graph between average accuracy and length of subsequence, with and without a default rule.

i.e. as we increase the length of subsequence, accuracy also increases and after a length of 35, accuracy ceases to increase. We also observe that though the accuracy increases as we increase the length of the subsequences, it is not a global pattern, particularly in the case of misuse-based approach (without a default rule). Therefore the length of the subsequence can be as short as 5 or as large as 40, we still can detect attacks without matching the whole process. The main point to observe here is that though normally, anomaly-based systems have good detection rate with a high false positive rate as compared to those based on misuse detection, our experiments show that anomaly-based IDS can have as low false positive rate as that of a misuse-based system with a very high detection rate. Also once the rules are computed, there is no further computation involved in classifying the processes. Only the subsequences of a process are matched against the rules for classifications. This makes the system very fast suitable for on-line detection.

5.7 *Conclusions*

Rough set theory is applied to many areas as one of many data mining and machine learning techniques. In this chapter, we propose the use of rough set theory in the area of intrusion detection to make it more suitable as on-line detection system. The main motivation behind using rough set for IDS is that boundary between normal patterns and abnormal ones is not always very sharp which leads to the ambiguity in the decision of the classifiers. Rough set theory is known for its capability to handle such type of data where uncertainty and vagueness is difficult to avoid. In rough sets, most of the operations involve comparing logical operators. Therefore it is also faster in decision making. The resources used in data collection, preprocessing of data and detection of intrusion are directly proportional to the number of features under consideration for each object. Obviously, in order to have a real time response from IDS, number of features should be minimized without affecting the classification power of the system. Rough set theory is capable of inducing rules that discard redundant attribute values. Knowledge representation is very simple and learning rate is very fast as compared to other techniques. Our study shows that it is possible to detect an attack by mare looking at some portion of the abnormal process. This reduces the size of the data to be processed and thus makes the subsequent computations much faster. The decision rules induced by rough set theory are easy to interpret and thus can be useful in further analyzing the events. We have tested our scheme by conducting experiments on DARPA'98 data. Empirical results justify our approach of making use of rough set for intrusion detection. As our future work, we intend to use the concept of *incremental learning* so that new rules can be learnt without retraining on whole data. We are also analyzing the IF-THEN rules to better understand the relationship among system calls to gain more insight about attacks. Our future work also includes to combine rough set method with other learning techniques, e.g. neural networks to propose a more robust IDS in terms of accuracy.

The described work in the chapter is published as:
Rawat S, Gulati V P, Pujari A K. (2005), "A Fast Host-Based Intrusion Detection System Using Rough Set Theory" To appear in: Transactions on Rough Sets, Vol 4, Springer.

CHAPTER VI

NETWORK TRAFFIC ANALYSIS AND WAVELETS FOR INTRUSION DETECTION

There is one thing even more vital to science than intelligent methods; and that is, the sincere desire to find out the truth, whatever it may be.

- Charles Sanders Pierce

6.1 Introduction

In previous chapters, we propose algorithms for host-based IDS. In this chapter, we investigate the applicability of wavelet theory in *network-based* IDS. The idea of going for NIDS is inspired by the fact that one of the most frequently occurring attacks over Internet is *denial of service* (DoS) attack (and, of course, its enhanced form, called DDoS), which is better analyzed by NIDS and wavelets are found to be useful in analyzing network traffic.

Considering the proliferation of *e-commerce* and our increasing dependence on Internet, a timely detection and prevention of DoS attack are of paramount importance. But, making the things more serious is the difficulty in stopping most of the DoS attacks. Therefore, it will be of great relief and beneficial to all Internet community if we can detect the ongoing DoS attack in nearly real time, which, in turn, implies that we should be able to process the data very fast. Therefore, the data should be as little in amount as possible, but must contain the patterns of attack, under consideration. In this chapter, we investigate the usability of some characteristics of network data to detect DoS attacks.

The *flooding attack* is one type of DoS attack. Therefore, under such type of attacks, if we count the number of packets or bytes during a fixed, but small, interval of time, we are likely to observe a sudden hike in the counts. Sometimes, it may happen that we may observe some pattern over a relatively long interval of time. If we collect the number of packets or bytes during last n minutes, this data set represents a time series. In other words, each value of count can be considered as the outcome of a function at different time. Therefore, such data can also be considered as a signal, or more precisely, a *non-stationary signal*, whose behavior at different time scales should manifest the *abnormalities* present in the data. We, therefore, adopt the signal analysis approach to detect the presence of *abnormalities* in network data.

It has been observed that Internet traffic is self-similar in nature. Self-similarity is the property that is associated with the objects whose structure is unchanged at different scales. It is pointed out [33] that the self-similarity of Internet traffic distributions can often be accounted for by a mixture of the actions of a number of individual users, and hardware and software behaviors at their originating hosts, multiplexed through an interconnection network. It has been observed that the traffic related to attacks, especially DoS attacks, is bursty in nature. Traffic that is bursty on many or all time scales can be described statistically using the notion of self-similarity.

In view of its multiscale framework and localization aspects (described in detail in the section 6.4), the wavelet technique is capable of being used for the analysis of scaling and local "burstiness" features of a function. In recent literature on wavelet based anomaly detection, the wavelet analysis has been used to describe and analyze data network traffic and its related problems like congestion, device failure etc [13]. The basic philosophy in wavelet based methods is that self-similarity is prevalent in the network traffic under normal conditions, and therefore can be considered as a signature for normal behavior. The loss of self-similarity, signifying a possible attack, can be taken as a deviation from normal behavior.

In network traffic analysis, detection of the presence of anomalies along with their locations and durations is an important issue for classifying the type of anomaly present in data. The existing approaches, especially wavelet based ones [66][80][147], detect efficiently the presence of anomalies. The present work aims at identifying the presence of anomalies along with their locations and durations in data.

Although the method presented herein is inspired by the method promoted in [66], it uses the relations present among the wavelet coefficients of self-similar functions in a different way to detect not only the deviation in the data from self-similarity, but also the locations at which the anomalies occur. We, later, extend our scheme to get some insight into the nature of the anomaly and the location it, by using singular value decomposition (SVD). The algorithm uses the properties of the right and left singular matrices, obtained in SVD of a matrix of local energies of wavelet coefficients, to determine the scales over which the data have possibly normal behaviour and locations at which the data have possible anomalous behaviour.

The chapter is organized as follows. Section 6.2 covers the necessary background to understand the approach, which includes *self-similarity*, *singular value decomposition*, and *wavelet theory*. In section 6.3, an algorithm to compute hurst parameter and its application to network-based IDS is shown, which is followed by the motivation for the present work in section 6.4. In section 6.5, we describe a new method to calculate Hurst parameter, using the multiscale approach and some experimental results on KDD dataset. Section 6.6 covers the extension of work defined in section 6.5 and describes a new SVD and wavelets based algorithm. Various observations and experimentation results on various datasets are also shown. The conclusion is provided in section 6.7.

6.2 Preliminary Background

Many objects in the real world, such as coastlines, are statistically self-similar: parts of them show the same statistical properties at many scales. Self-similarity is a typical property of fractals. It also has important consequences for the design of computer networks, as typical network traffic has self-similar properties. For example, in telecommunications traffic engineering, packet switched data traffic patterns seem to be statistically self-similar. This property means that simple models using a Poisson distribution are inaccurate, and networks, designed without taking self-similarity into account, are likely to function in unexpected ways. In this section, we present the definition of self-similarity and the basic concepts associated with it, which is followed by a detailed description of wavelets. Wavelets have been found to be useful for many data mining tasks, like *dimensionality reduction* [1][153] and *clustering* [174] etc. A good overview on the application of wavelets to data mining has been given in [119].

6.2.1 Self-Similarity

Fractals usually possess what is called self similarity across scales. That is, as one zooms in or out, the geometry/image has a similar (sometimes exact) appearance. Formally we can define self-similarity as follows [66]:

Definition: A process or function $\{f(t) : t \in (-\infty, \infty)\}$ is said to be self-similar with self-similarity parameter H, if and only if $\{c^{-H}f(ct) : t \in (-\infty, \infty)\}$ and $\{f(t) : t \in (-\infty, \infty)\}$, $\forall \quad c > 0$ and $t \in (-\infty, \infty)$ have same distributions, which is referred to as the scaling property of the process f. It can be noted that every function of the form $f(x) = nx,\ n \in \mathbb{N}$ represents a self-similar function. To illustrate the self-similarity of a function, let us take the following data into consideration, where each data point can be considered as function's value at different time t. For simplicity, we take $H = 1$ and

1,2,2,4,2,4,4,8,2,4,4,8,4,8,8,16,2,4,4,8,4,8,8,16, 4,8 8, 16,8,16,16,32,
2,4,4,8,4,8,8, 16,4,8,8,16,8,16,16,32,4 8,8,16,8,16,16,32,8,16,16,32,
16,32, 32,64,2,4,4,8, 4,8,8 16,4,8,8,16,8,16,16,32,4,8,8,16,8,16,16,32

Table 33: Example of Self-Similar data

$c = 1, 2, 4$ in the definition of self-similarity. The following figure 22 shows the data at different scales (different values of c).

The high variability in data network traffic is due to the Long Range Dependence (LRD) property of the traffic processes.

Definition: The LRD property means that auto-correlation function $r(k)$ of a wide-sense-stationary process X_n slowly decreases according to a power law such as $r(k) \sim c_r k^{-(2-2H)}$ as $k \to \infty$, where $c_r > 0$ and $H \in (0.5, 1)$ [194].

This mostly means that the data $f(t)$ remain strongly correlated over very large scales of time (or space) and, therefore, the system which produce them present long memory behaviors. If the correlation holds over a short scales of time, the function is said to have *short range dependence* (SRD). The parameter H is called Hurst parameter. For a general self-similar process, the parameter H measures the degree of self-similarity. For random

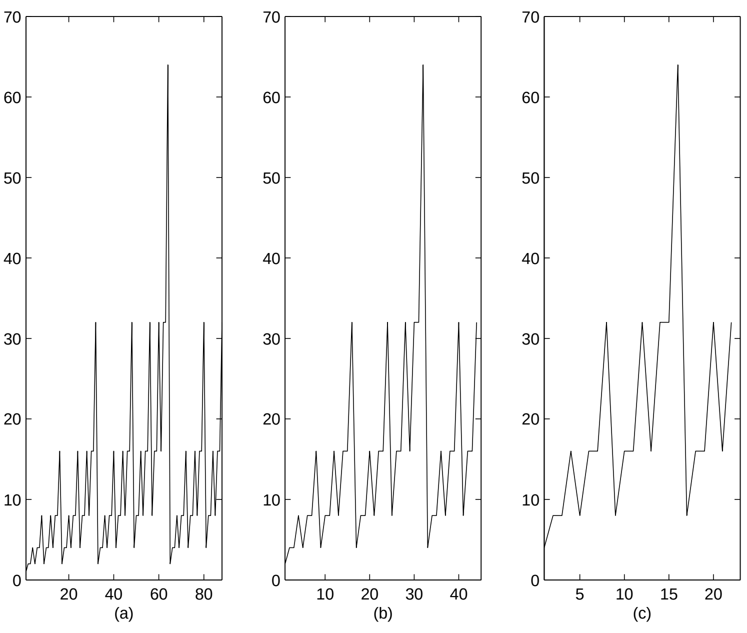

Figure 22: Example of self-similar data. (a) Data at $c = 1$, (b) Data at $c = 2$, (c) Data at $c = 4$.

processes suitable for modeling network traffic, H is basically a measure of the speed of decay of the tail of the autocorrelation function. If H lies between 0.5 and 1, the process is LRD and if it lies between 0 and 0.5, the corresponding process is said to have Short Range Dependence (SRD). Hence H is widely used to capture the intensity of LRD. In traffic modeling, however, the term 'self-similar' is usually used to refer to the asymptotically second order self-similar process [194] i.e the LRD process. Hence, in the study of anomaly behaviour of traffic data, any deviation of H from 0.5 to 1 range signifies the presence of anomaly in the data [2][66]. Throughout our study, we use this observation for the detection of intrusion present in the data. From now on, we consider f to be a finite variance self-similar process with the self-similarity parameter $H \in (0.5, 1)$. In mathematical terms, we, however, treat the function f as being a self-similar finite energy function possessing finite (support) duration.

6.2.2 Singular Value Decomposition Revisited

In the coming sections, we will be making use to some properties of SVD, which have not been mentioned in the section 3.2.2. Therefore, we revisit SVD in this section to discuss some other properties.

The SVD of an $m \times n$ matrix $A = [c_1 c_2 \dots c_n]$, with c_i being the columns, is the decomposition of A into the product of three matrices, as follows [67]

$$A = U\Sigma V^T = \sum_{k=1}^{r} \sigma_k u_k v_k^T, \tag{34}$$

where, the orthogonal matrices $U^T = [u_1 u_2 \dots u_m]$ and $V^T = [v_1 v_2 \dots v_n]$ are the left and right singular matrices of A respectively. The matrix Σ is the diagonal matrix whose first r (the rank of AA^T) diagonal entries are $\sigma_1, \sigma_2, \dots, \sigma_r$ (sungular values of A), and the rest are zeros. We firstly observe the following results, which we use in the later sections to identify the scales/locations over which the network data are normal/abnormal.

Property 1: For any two columns, say c_i and c_j, of A,

$$c_i = \lambda c_j \quad \text{if and only if} \quad v_i^{tr} = \lambda v_j^{tr}, \tag{35}$$

where, v_i^{tr} and v_j^{tr} are respectively the first r elements of v_i and v_j in vector form, and λ is any scalar.

Proof: From equation (34), one can write the columns c_i and c_j as

$$c_i = U\Sigma v_i \quad \text{and} \quad c_j = U\Sigma v_j.$$

Therefore, we have

$$c_i - \lambda c_j = U\Sigma(v_i - \lambda v_j).$$

Since the matrix U is invertible and as the first r elements of the diagonal matrix Σ are nonzero, we have

$$c_i - \lambda c_j = 0 \iff U\Sigma(v_i - \lambda v_j) = 0 \iff \Sigma(v_i - \lambda v_j) = 0 \iff v_i^{tr} = \lambda v_j^{tr}.$$

$\blacksquare$

Note that the factors, by which the pairs of columns c_i and c_j, and v_i^{tr} and v_j^{tr} are linearly dependent, are same. Similarly, when any two rows of A are in proportion, one can prove (using A^T in place of A in proof) that up to the rank of A, the corresponding rows of U are also in the same proportion.

Using the orthogonality of U and singular values σ_i of A, one may compute $\|c_i\|$ (distance from origin) of the column c_i as follows

$$\|c_i\|^2 = \sum_{k=1}^{rank(A)} (\sigma_k v_{k,i}^{tr})^2 \tag{36}$$

Similarly, using the orthogonality of V, one may compute the distance of row r_i of A as follows

$$\|r_i\|^2 = \sum_{k=1}^{rank(A)} (\sigma_k u_{k,i}^{tr})^2 \tag{37}$$

6.2.3 Wavelets Analysis

A wavelet is a "little wave" that is both localized and oscillatory. The representation of a function in terms of wavelet basis functions (generated by dyadic scaling and integer translates of the wavelet) involves a low frequency block containing the "identity" of the function and several high frequency blocks containing the visually important features or "flavours" (such as edges, lines or spikes). Therefore, the wavelet transform is expected to provide economical and informative mathematical representation of many objects of interest [133]. A function $f \in L^2$ (the space of all finite energy functions defined over the real line $\mathbb{R} = (-\infty, \infty)$) has the following wavelet representation

$$f = \sum_{k \in Z} c_{J,k}\phi_{J,k} + \sum_{j \geq J;k} d_{j,k}\psi_{j,k} = \sum_{j,k \in Z} d_{j,k}\psi_{j,k} \tag{38}$$

where $\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi(2^j t - k)$, $c_{J,k} = \langle f, \phi_{J,k}\rangle$ and $d_{j,k} = \langle f, \psi_{j,k}\rangle$ with $\langle .,. \rangle$ being the standard L^2 - innerproduct operation. Ingrid Daubechies [40] has given a procedure for the construction of compactly supported, sufficiently regular and orthonormal wavelets. A detailed mathematical description of wavelets is given in Appendix A.

Due to the easy accessibility of many software packages that contain fast and efficient algorithms to perform wavelet transforms, wavelets have quickly gained popularity among scientists and engineers working on both theoretical issues and applications. Wavelets have been widely applied in computer science areas such as image processing, computer vision, network management and data mining.

Wavelets have many favourable properties, such as compact support. The property of wavelets being compactly supported implies the localization feature of them. The main advantage of this feature is that the presence of local error (noise) in the data reflects local changes in wavelet coefficients, unlike the Fourier technique wherein the local change in data has global effect on the Fourier coefficients. This feature along with multiresolution feature is widely being used in image/signal/pattern analysis [162]. Although one can use different types of wavelets in IDS, through out this chapter, we use orthonormal Daubechies wavelets. Besides the stated properties, wavelet bases possess other properties like zero moments, hierarchical and multi resolution framework, and 'decorrelated' coefficients. These features could provide considerably more efficient and effective solutions to many practical problems.

6.3 *Wavelets for Network-Based IDS*

In this section, using the properties of wavelet bases and self-similar functions, we make some observations on the methodology described in [2][66] for the detection of anomalies in the data. The description of the technique can be found in the section 2.7 on page 60. Under the scheme of Gilbert, the plot between *log of energy* and scales is drawn. The scales over which the plot is straight line are determined first to identify the scale interval over which the self-similarity possibly holds. Then, from the slope of the line, i.e., $-(2H+1)$, H is computed. If the Hurst parameter H computed falls between 0.5 and 1, the data are self-similar LRD process. In the event of either H not lying between 0.5 and 1 or no straight line behaviour in the 'energy-scale' plot, the data are said to possess anomaly.

Remark.1: It may be noted that the energy plots can be linear for some functions and, in spite of it, the corresponding function f in question may not satisfy the scaling property. For example, suppose f is a polynomial of some degree N over the interval $[-2^J, 2^J]$ and 0 outside it, for some positive integers J, N. When we use the wavelet basis possessing zero moments up to order N-1 (i.e., $m \leq N-1$ in equation (52)), the energy plot is straight line between the levels 0 and J, and the slope of the line, as dictated by the choice of N, can take arbitrary values. This is, of course, a hypothetical function, taken as an example to justify that the straight line behaviour of the 'energy-scale' plot does not necessarily imply the self-similarity.

Remark.2: Suppose f is a self-similar and LRD stationary process. The explanation given in the foregoing paragraphs concludes that the energy plot is a straight line, and the H determined from the slope of the energy plot falls in $(0.5, 1)$. It may be noted that the converse is not necessarily true, due to the following reason: suppose the energy plot of a function g is straight line over scale interval $[j_1, j_2]$ with the corresponding H falling in the desired interval. Then g need not be self-similar, as follows. So for any noise function $\eta(x)$, (for example, $\eta \in V_l$ for $l < j_1$ or $\eta \in W_l$ for $l > j_2$), that is orthogonal to $\psi_{j,k}$, $\forall k$ and $j \in [j_1, j_2]$, we have

$$< f + \eta, \psi_{j,k} >=< f, \psi_{j,k} > + < \eta, \psi_{j,k} >=< f, \psi_{j,k} >$$

Hence, the energy plots of f and $g = f + \eta$ are exactly same. But the function $f + \eta$, due to the presence of noise factor η, may not be a self-similar process. This problem, however, can be overcome by replacing f with $f - Proj_{V_{j_1}} f - \sum_{j \geq j_2} Proj_{W_j} f$. In case of no noise factor, the projection factors become zero, and hence up to the additive noise factor, we can conclude from the straight line behavior, the presence of self-similarity in data. Here, $Proj_X f$ stands for the projection of f onto X.

6.4 *Motivation for the Present Work*

Basic idea in the wavelet based approach is that, when data satisfy scaling property, the corresponding wavelet coefficients and hence the energies of them at different levels also satisfy the same scaling property (with different exponents). Analyzing the exponents of scaling property of energies at different levels, we conclude if the data satisfy the scaling property.

Let α and β be two vectors of same dimension. Let their distances (from origin) $\|\alpha\|$ and $\|\beta\|$ satisfy the relation: $\|\alpha\| = \lambda\|\beta\|$ for some positive number λ. Geometrically, for any α and β lying respectively on the surfaces of the spheres around origin of radii $\|\alpha\|$ and $\|\beta\|$, the relation $\|\alpha\| = \lambda\|\beta\|$ holds. Consequently, $\alpha = \lambda\beta$ may not, in general, hold.

The energies computed (without normalization factor) in the previous section are squares of distances of wavelet-coefficient-vectors from origin at different levels. Since the wavelet coefficients at different levels are representation coefficients of orthogonal components of a given data (hence they appear to be unrelated), and in view of use of undecimated wavelet transform, from the foregoing explanation, one may be tempted to say that the conclusion of scaling property of data from that of energies at various levels may at times be inappropriate. That is, if α and β are d^{j+1} and d^{j} (j and $j+1$ level coefficients) respectively, then $\|\alpha\|^2$, $\|\beta\|^2$ and λ are $N_j E_{j+1}$, $N_j E_j$ and $2^{-\frac{2H+1}{2}}$ respectively, and the relation $E_{j+1} = 2^{-\frac{2H+1}{2}} E_j$ may not imply $d_{j+1,k} = 2^{-\frac{2H+1}{2}} d_{j,k}$.

In addition to the aforestated point, in anomaly detection, answering accurately the question "when and where" is important. However, to our knowledge, existing methods detect just the presence of anomalies, and not their locations. In view of some anomalies (like flash crowd attacks) possessing long durations while some others (like network failures) possessing short durations, it may as well be interesting to identify the duration of anomalies. We, therefore, aim at detecting in data the presence of anomalies, their locations and durations using multiscale transform and singular value decomposition.

6.5 *New Multiscale Algorithm for Network Traffic Analysis*

The stated objective may be achieved by using the relation, given by equation (**??**),

$$H = \frac{1}{2}\left[\frac{2}{n}log_2\left(\frac{d_{m,k}}{d_{m+n,k}}\right) - 1\right]. \tag{39}$$

The Hurst parameters H computed for different m, n and k, reveal not only the scale interval over which H falls in the desired range, but also the time instances where H goes out of range in the event of presence of anomaly in data. We believe that this observation helps us detect not only the presence of anomaly in the data, but also the location of it.

Though the precise mathematical relation shown in equation (**??**) ensures one-to-one relation between the coefficients at all pairs of levels of self-similarity, more often than not, it does not match with what we see in practice. The reason for this is: in actual computations, when dealing with compactly supported wavelet basis functions, we use finite number of coefficients at every (finite) level and as we increase the level, number of coefficients involved becomes doubled. Consequently different levels have different number of coefficients. In view of it, taking ratios of coefficients of two levels as shown in equation (39) is seemingly unjustified. To get rid of this problem, in our simulation work, we use $f \star \overline{\psi}_j(2^{-j}k)$ to compute $d_{j,k}$ (with $\overline{\psi}_j(x)$ being $2^j\psi(-2^jx)$). At every level j, we collect appropriate coefficients from $f \star \overline{\psi}_j$ and then take the ratios as shown in equation (39). Leaving it aside, after computing $f \star \overline{\psi}_j$ at different j and treating them as coefficients, we take ratios of them at different j. Although it has no justification, we observe that both the approaches give similar results.

6.5.1 Experimental Results on KDDcup'99 Dataset

In 1998, DARPA, together with Lincoln Laboratory at MIT, released the DARPA 1998 data set [35] for the evaluation of IDS. The data set consists of seven weeks of training and two weeks of testing data. There are a total of 38 attacks in training and testing data. The processed version of DARPA data, containing only the network data (Tcpdump data), is termed as KDD data set [89]. There are approximately 5 million records in training set and 0.3 million records in the testing set. Each of these records is described with 41 attributes, wherein 34 are numeric and 7 are symbolic. This set of attributes includes general TCP features like bytes/sec, flags, duration etc and derived features like the-same-host features and the-same-service features. A detailed description of these features can be found in [115]. It has been observed [6] that the DARPA data exhibit self-similarity within certain interval of time. As KDD data set is derived from DARPA data set, out of curiosity, we choose KDD data set for our experimentation. Recently, it is shown by Sabhnani and Serpen [160] that KDD data set is not suitable for misuse-based IDS as the distribution of attacks in training and testing data is not same. The portion of new attacks, in the R2L and U2R categories, in testing data is more than the known attacks. Therefore any misuse-based IDS will show poor detection rate. Ours is an anomaly-based approach, consequently the choice of this data set is not affected by the shortcomings of KDD set as pointed out in [160]. We test our approach on the 10% of the KDD set, which is provided

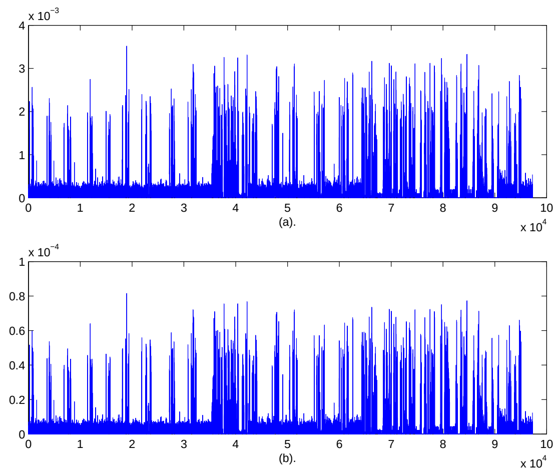

Figure 23: The changes in H values of the normal data

with the full set. This set consists of almost all the attacks present in DARPA data set with a total of 4,94,020 records. Out of these records, there are 97,277 normal records in the data set which we have considered in the beginning part of the data by sorting out the data suitably.

We have carried out our simulation work using nearly symmetric orthonormal wavelet ('sym6') of MATLAB package. Using equation (39), we have computed the Hurst parameter H at different levels. The H values computed, remaining close to 0.8 and hence falling in the interval $(0.5, 1)$, reveal that self-similarity of the data is present between the levels 1 and 4. In figure 23, we have displayed the gradient of H values at the levels 3, 4 to demonstrate how H changes within a given level. Our motive behind taking gradient of H values is to notice the abrupt changes in H as it may signify the presence of anomaly in the data. To show this concept empirically, we choose the Neptune attack from the KDD data set, which is a DoS attack sending large number of SYN packets to target host to exhaust its buffer. As this attack never establishes the session, many bytes become almost zero in each connection attempt. Column 2 of KDD set indicates this feature. Under normal condition, the behavior of column 2 in terms of H is shown in figure 23. It can be seen that the changes in H are of very small order.

To test the detection ability of our approach, we introduce many sessions of Neptune attack in the end of the data. The H-gradient plot of this data is shown in figure 24.

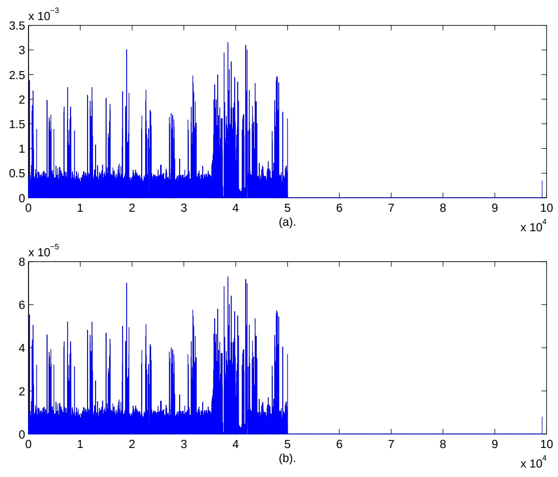

Figure 24: The changes in H values in the presence of Neptune attack

It is clearly visible, in figure 24, that the variation in H corresponding to the anomalous data, (i.e., in the right half of the figure 24), is almost zero. However, the actual values of H go out of the desired range. This is happening because, as explained above, for a long period of time, the attacker is just sending SYN packets without establishing the connection, resulting in similar values for H at all levels of significance (hence the changes in H values are almost zero). We also carry out the similar experiments with smurf attack, present in the KDD data set. Under this attack, the victim receives a large number of ICMP reply (echo) within a short period of time. This hampers the performance of victim to respond it properly to other requests from genuine senders. This activity is captured in the KDD set in the column "count", which counts the number of connections to the same host as the current connection in the past two seconds.

On the basis of above experiment, it can be inferred that the proposed approach is capable of detecting the attacks wherein the burstiness is present in the traffic. In spite of this, the algorithm is computationally expensive. Besides, it may so happen that wavelet coefficients may become zero so that the ratio, as shown in equation (39), takes indeterminate form. To get rid of this one has to treat such cases specially. To get some more insight into the nature of the network anomalies, in the next section, we explore a new technique that is free from the problem stated above.

6.6 A New Wavelet-SVD Hybrid Algorithm for Anomaly Detection

Although the observations made in the previous sections, may help us detect both anomalies and their locations, it is computationally very expensive and hence can be of little use in actual implementations.

Motivated by the observations made in the previous sections, in the following section, we define a matrix whose elements are, in a way, local energies of wavelet coefficients, and using the properties of SVD observed in section 6.2.2, we present a wavelet-SVD hybrid algorithm for the analysis of network traffic data. The reasons for using SVD are: 1). information about the number of linearly independent (and hence, dependent) rows/columns present in A can be obtained from the number of singular values of A, 2). energies of wavelet coefficients at various levels may be computed using few (effective) singular values, as shown in equations (36) and (37), 3). properties of left and right (i.e., U and V) singular matrices can be used to determine the locations and scales of anomaly in data respectively.

Let us define the matrix A with local energies, without normalization factor, as its elements

$$A = A_{j,J} := \frac{1}{N} \begin{pmatrix} \sqrt{\sum_{k=1}^{M} d_{j,k}^2} & \sqrt{\sum_{k=1}^{M} d_{j+1,k}^2} & \cdots & \sqrt{\sum_{k=1}^{M} d_{J,k}^2} \\ \sqrt{\sum_{k=M+1}^{2M} d_{j,k}^2} & \sqrt{\sum_{k=M+1}^{2M} d_{j+1,k}^2} & \cdots & \sqrt{\sum_{k=M+1}^{2M} d_{J,k}^2} \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{\sum_{k=N-M+1}^{N} d_{j,k}^2} & \sqrt{\sum_{k=N-M+1}^{N} d_{j+1,k}^2} & \cdots & \sqrt{\sum_{k=N-M+1}^{N} d_{J,k}^2} \end{pmatrix}, \qquad (40)$$

where j and J are finer and coarser resolution levels of wavelet transform respectively, and N and M are the total number of (undecimated) wavelet coefficients and number of wavelet coefficients used in the computation of local energy respectively. The sum of squares of a column elements of A represents the energy of wavelet coefficients at a given level, while the same of a row represents the total energy of wavelet coefficients in given time duration over all scales used in defining A. Due to the reasons stated in sections 6.3 and 6.4, the columns in A on the left and right sides represent higher and lower frequency levels respectively.

The choice of j and J pair may be made based on the requirement. For example, if the data are collected at 1min step size and if we are to analyze the presence of anomalies of 1-4 hour duration, j and J may be set to 6 ($2^6 \approx 1hour$) and 8 ($2^8 \approx 4hours$) respectively. As M controls the extent of local energy, the choice of M may have some bearing on the duration of anomalies present in data (discussed in more detail in section 6.6.1).

Ideally, if the data are self-similar over the scales, say $j, j+1, \ldots, J$, as is considered in A, from the scaling property of data, one can write A as

$$A = \frac{1}{N}\begin{pmatrix} \sqrt{\sum_{k=1}^{M} d_{j,k}^2} & 2^{-\frac{(2H+1)}{2}}\sqrt{\sum_{k=1}^{M} d_{j,k}^2} & \ldots & 2^{-\frac{(2H+1)(J-j)}{2}}\sqrt{\sum_{k=1}^{M} d_{j,k}^2} \\ \sqrt{\sum_{k=M+1}^{2M} d_{j,k}^2} & 2^{-\frac{(2H+1)}{2}}\sqrt{\sum_{k=M+1}^{2M} d_{j,k}^2} & \ldots & 2^{-\frac{(2H+1)(J-j)}{2}}\sqrt{\sum_{k=M+1}^{2M} d_{j,k}^2} \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{\sum_{k=N-M+1}^{N} d_{j,k}^2} & 2^{-\frac{(2H+1)}{2}}\sqrt{\sum_{k=N-M+1}^{N} d_{j,k}^2} & \ldots & 2^{-\frac{(2H+1)(J-j)}{2}}\sqrt{\sum_{k=N-M+1}^{N} d_{j,k}^2} \end{pmatrix}. \tag{41}$$

Note that in A, all rows are scalar multiples of one row, and all columns are scalar multiples of one column. Hence the local energy matrix A contains one independent row and one independent column. From equation (41), one may observe that the energy of i^{th} row, i.e., r_i of A satisfies the relation:

$$R_i^2 := \frac{\|r_i\|^2}{\sum_{k=1+(i-1)M}^{Mi} d_{j,k}^2} = \sum_{k=j}^{J-j} 2^{-(2H+1)k} = \text{constant independent of } i. \tag{42}$$

In the event of presence of anomaly in data, this relation, however, may not hold for the rows that contain anomalous behaviour either in parts or in full. We make use of this point, and the algorithm that we propose has the following methodology:

- Using equations (35) and (36), the properties of matrix V (right singular matrix of SVD of A) and 'energy-scale' plot, we identify the consecutive and maximum number of linearly dependent columns of A for which the Hurst exponent H lies between 0.5 and 1.

- Using equations (37) and (42), we determine the linearly independent rows of A.
- Integrating the information obtained from linearly dependent columns and linearly independent rows of A, we make our analysis for anomaly detection.

Given data, to identify if it is normal, we generate a matrix with arbitrary scales, say from 1 to J, i.e., $A = A_{1,J}$ (J used is a generic symbol here, if we are to determine the anomalies of specific duration, we use j and J accordingly as needed). Using the SVD of A, we, first, identify the scales at which the columns are linearly dependent, these columns signify the scales over which the data are likely to be self-similar. The identification of the dependent columns is, nevertheless, made using the relations shown in equations (35) and (36), as follows: computing first the energies of first r elements of all columns of V^T, and then using the 'energy-scale' idea described in section 6.5. Although we also use the 'energy-scale' idea, we are less likely to face the problem described in the second paragraph of section 6.4. This is because, we deal with a small number, r, of elements of rows of V.

After identifying the linearly dependent columns, we determine the linearly independent rows of A using equation (42) as follows: We compute R_i for each row of A. The rows that are linearly dependent have nearly same R_i values, and the rows that are linearly independent will have indifferent R_i values. Guided by this intuition, we designate the rows that have larger deviations from the mean of R_i (i.e., $R_i^d = |R_i - mean\{R_i\}|$, called Deviation score) as being linearly independent.

Since the rows restricted to the linearly dependent columns are also linearly dependent, having identified linearly independent rows, we interpret those parts of the independent rows that fall outside the submatrix possessing identified linearly dependent columns of A as being the *pockets of anomalies*. If the identified independent rows are present in consecutive positions, we may conclude that the anomaly is of long duration. The very nature of wavelet bases (due to their zero moment properties) implies that the anomalies of short duration (high frequency bursts) are in general better captured by high frequency parts (in such cases, columns on the left of A will become linearly independent). On the other hand, anomalies of long duration are better captured by lower frequency coefficients (in such cases, columns on the right of A will become linearly independent). Consequently, from the *pockets of anomalies*, we can figure out the type of anomalies.

Let us suppose that the columns d to $d+m$ of A are the columns that have normal behaviour. If the response of wavelet ψ lies in $[a,b]$, then $\psi_{j,k}$ has its response present in $[\frac{a+k}{2^j}, \frac{b+k}{2^j}]$. If the *pocket of anomaly* on the row r_i is $r_i(1:d)$, then it may be interpreted that the data have anomalous behaviour in $\left[\frac{a+(i-1)M+1}{2^j}, \frac{b+iM}{2^j}\right]$ for $j \neq d, \ldots d+m$. The pseudocode of the algorithm is given in figure 25.

```
given data, choose scales $j, J$ as needed, and the number $M$ for extent of local energy
compute wavelet transform of data between scales $j$ and $J$;
construct the matrix $A$ of local energies, as shown in equation (40);
    for each scale between $j$ and $J$
        for each block of length $M$ of wavelet coefficients
            $A$(block number, scale) $\leftarrow$ Calculate the local energy;
        end
    end
calculate SVD of $A$ as $A = U\Sigma V^T = \sum_{k=1}^{rank(A)} \sigma_k u_k v_k^T$;
identify the number of significant singular values $\rho$ of $A$;
truncate the column size of $U$ and $V$, and diagonal size of $\Sigma$ to $\rho$;
find $d$ and maximum $m$ s.t. the columns $v_d$ to $v_{d+m}$ are dependent and $H \in [0.5, 1.0]$, using equations (35) and (36) and 'energy-scale' plot;
    for each row $r_s$ of $A$
        Using equations (37) and (42), compute $R_s$;
    end
compute the mean $R_{mean}$ of all $R_s$;
    for each $R_s$
        Compute the deviation score: $R_s^d = |R_{mean} - R_s|$;
    end
select the first $\rho$ number of rows of highest $R_s^d$, say, $r_{i_1}, \ldots, r_{i_\rho}$;
Output: pockets of anomaly on each of $r_{i_l}$ are $r_{i_l}(1:d-1)$ and $r_{i_l}(d+m+1:J-j+1)$
  If $i_1, i_2, \ldots i_\rho$ are consecutive, anomaly in data is of long duration. Otherwise, its presence is sparse
  If the pockets of anomaly on rows $r_{i_l}$ are on the left of $r_{i_l}$, anomaly is bursty in nature.
```

Figure 25: Algorithmic representation of the proposed scheme

6.6.1 On the Choice of M

When data are constant in some portions, the zero mean property of wavelets implies that the corresponding wavelet coefficients become zero, and hence the local energies can possibly become zero. Since the definition of R_i in equation (42) involves the presence of local energies in the denominator, M should be taken in such cases so large that the local energies on the first column of A in equation (40) are nonzero. For example, in the detection of *DoS* type of anomaly, suppose the data set consists of number of packets during the last , say, 1 minute time interval. This represents a time series wherein each entry corresponds to number of packets during the last 1 minute. Let us assume that in order to flood a system, it is required to send x packets per minute (x is reasonably large) for 5 minutes. In this case, if we choose M equal or less than 5, then it may happen that all the entries in the window, corresponding to the attack, may be x and thereby making corresponding wavelet coefficients zero, as mentioned above. Therefore, to be on safer side, we should take M greater than 5. On the other hand, smaller values of M may enable the algorithm to identify the location of anomaly precisely.

6.6.2 Simulation Results with New Hybrid Method

We use first order Daubechies (i.e. Haar) wavelet [40] throughout our simulation work. Strictly speaking, the data set that we use in our simulations does not have natural decay to zero on both ends, which can be a source of problem (border effects due to Gibb's phenomenon) when the implementations are carried out via the frequency domain. The number of coefficients that get affected is determined by the width of wavelet basis being used. One effective way to overcome this problem is to use the wavelets that are adapted to bounded domains [40]. The other way is to discard those wavelet coefficients on both ends that get polluted by border effects, which is relatively convenient computationally. In our simulations, as we use lower order wavelet, we overcome the problem by discarding 1 or 2 wavelet coefficients on both ends.

To begin with, we show our results for the test data shown in table 34, whose wavelet transform at first 5 levels is shown in figure 26. The reason for using this small dataset is to visualize the results manually. As our test data set has 88 elements, we define A with 1 to 6 levels (since, $2^6 < 88 < 2^7$ and as 6 is the number of possible levels requiring no zero padding in conventional wavelet transform, we set J to 6). As an example, we use M=8 in defining A, hence A is a matrix of size 11×6.

1,2,2,4,2,4,4,8,2,4,4,8,4,8,8,16,2,4,4,8,4,8,8,16, 4,8 8, 16,8,16,16,32,
2,4,4,8,4,8,8, 16,4,8,8,16,8,16,16,32,4 8,8,16,8,16,16,32,8,16,16,32,
16,32, 32,<u>64</u>,2,4,4,8, 4,8,8 16,4,8,8,16,8,16,16,32,4,8,8,16,8,16,16,32

Table 34: Self-Similar test data

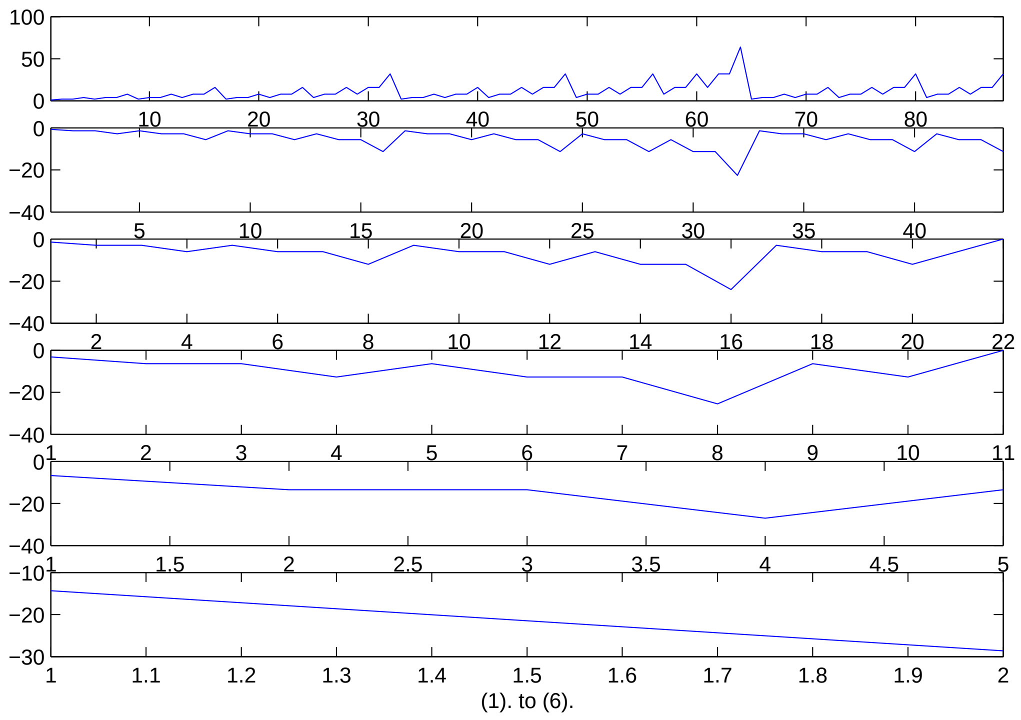

Figure 26: (1). Plot of the data shown in table 34, (2). to (6). wavelet transform at first 5 levels.

After applying SVD on A, we decimate those singular values that have magnitude below some tolerance (say 0.5). One may observe that the effective rank of A is 2. Using the first two elements of columns of V^T, we compute the energies of columns of V^T and then plot the energies, shown in figure 27(b), to determine the linearly dependent columns of V^T for which H lies between 0.5 and 1. For the data shown in table 34, we observe that columns 2 to 6 are linearly dependent, while the column one is independent of others, justifying that the effective rank of A is 2. In figure 27(a), we have shown the 'energy-scale' plot computed using the columns of A. One may observe that both the plots in figure 27 have similar behaviour, and hence using the first few elements on the columns of V^T, one can identify the scales over which the data have normal behaviour.

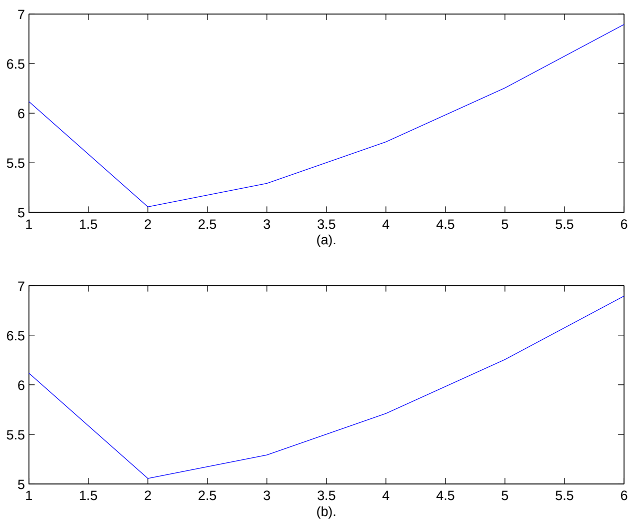

Figure 27: 'Energy-Scale' plots of energies computed using (a). A, and (b). $V(1:2, 1:6)$; showing that plots have similar behaviour and are almost straight lines between the scales 2 and 6.

To identify linearly independent rows of A, using equation (42), we compute R_i, which are 9.6055, 8.1344, 4.9051, 4.8581, 5.2151, 4.8581, 1.6005, 1.6945, $\underline{27.1086}$, 4.8581 and 1.6005. For our test data, the ninth and first rows have maximum deviations from the mean of R_i, and hence those rows are linearly independent. As the matrix A has effective rank 2, we are interested in finding two linearly independent rows, which in the present case turn out to be ninth and first rows. As the column elements 2 to 6 on these rows are nearly in proportion, the first element of these rows must have anomalous behaviour. It may be noted that the 64^{th} data element, underlined in table 34, has value 64, which is maximum and the value next to it is 2. Since the change in data values to the right of 64 is more significant, our data set has a rapid change of very short duration in the beginning of ninth bin (M=8, 65^{th} position means ninth bin). The linear independence of first column in A and the presence of local burst in data justify our point that the linear independence of left and right columns signifies the presence of anomalies of short and long durations respectively. It may be noted that, when running the algorithm on data set in table 34 taken from last to first, the the fourth row has maximum deviation score (figure 28). It justifies the presence of anomaly in fourth bin as the anomaly in data(1,end:-1:1) is located at 25th position.

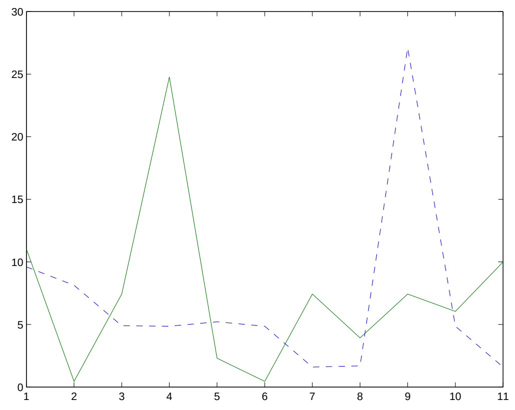

Figure 28: Deviation scores R_i^d for the data shown in table 34. Dashed line is for data and continuous line is for data taken from end, i.e., data(1, end:-1:1); the deviation scores in both cases guarantee the accuracy with which the present method detects the presence and location of anomaly.

6.6.3 Results on Kddcup'99 dataset

In this section, we provide results on Kddcup dataset. We select the *back* attack for our experimentation. On analyzing the data, we observe that under the *back* attack, there is sudden increase in the bytes from source to destination (column 2 of kdd dataset). We select some normal instances and then mix the instances of back attack in the data (total data size is 1100). The data is shown in figure 29. The linearly independent rows corresponding

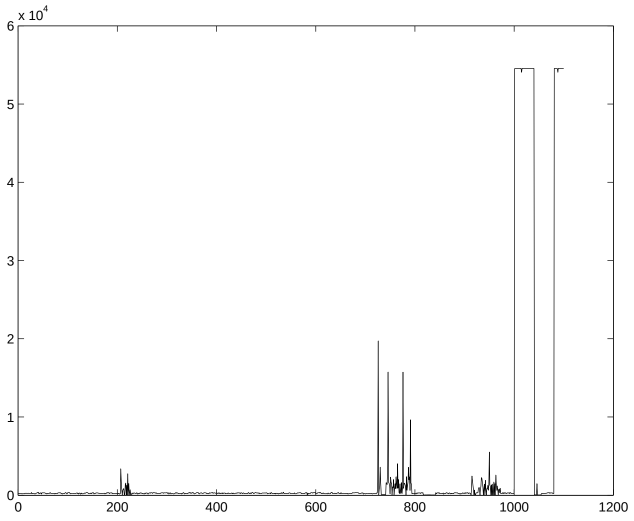

Figure 29: Kddcup'99 dataset - normal and *back* attack instances. The spikes at points 1000 and 1081 depict the presence of back attack.

to highest deviation are - 42 41 44 29 17 18 1 27 21. It should be noted that with $M = 25$, the attack instances (from 1001 to 1040 and from 1081 to 1100) falls in rows 41, 42 and 44 which is correctly captured by the method. Figure 30 shows the deviation of LI rows at the point of anomaly.

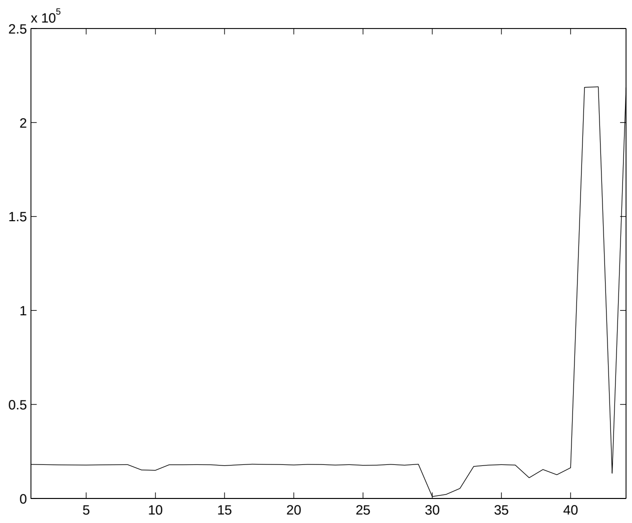

Figure 30: Deviation R_i^d of LI rows from the mean value in Kddcup'99 dataset - normal and *back* attack instances. The spikes at points 40 and 44 conform to the back attack data in figure 29.

6.6.4 INFINET Data Results

INFINET (Indian Financial Network) is a CUG (close user group) network, managed by IDRBT [84]. This network is primarily used by various banks for their inter- and intra-bank activities. In statistical terms, being a CUG network, its behavior is very much regular. Therefore, it should be a valid assumption to consider INFINET traffic as self-similar. The network is connected by Cisco routers at various locations. We, therefore, use NetFlow tool to collect the packet counts as our experimental data. We collect the data for one week with an interval of size 5 minutes and a sliding window of size 3 minutes. As the network is CUG, the possibility of any attack is very less. Therefore, our simulation results are expected to show the presence of self-similarity only. However, at some points, a sudden hike in packet counts is detected. After consulting the network engineer, we have found that those sudden spikes are due to the failure of one of the interfaces of the router.

Therefore, the traffic of the failed interface is diverted to other interface, making thereby the number of packet from that interface very high for some time. Such type of anomaly is also termed as *flash crowd* anomaly.

We have implemented our algorithm on INFINET data shown in figure 31. Setting M to 8 and taking 11 admissible levels, we have generated a matrix of size 269×11, which happens to be a full rank matrix. When running the programme using usual 'energy-plot' idea,

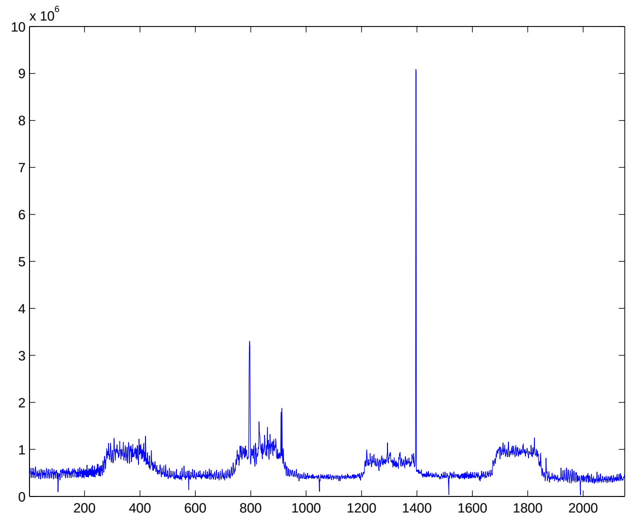

Figure 31: INFINET data

we have observed that the data have normal behaviour (straight line in energy-scale plot with the H falling in $[0.5, 1]$, shown in figure 32) between the scales 2 and 11. However, the effective rank of A being 11 implies that all columns in A are linearly independent, which is justifying our observation made in the motivation part that the conclusion of scaling property of data from that of energies may at times be inappropriate. When analyzing the deviation scores, we have found that the bins 176, 101 have maximum deviation scores among others, shown in figure 33, which indicates that the positions around 1400 and 800 in INFINET data in figure 31 have anomalous behaviour. There are other 9 rows in A that have significant deviation scores, which we are not analyzing here.

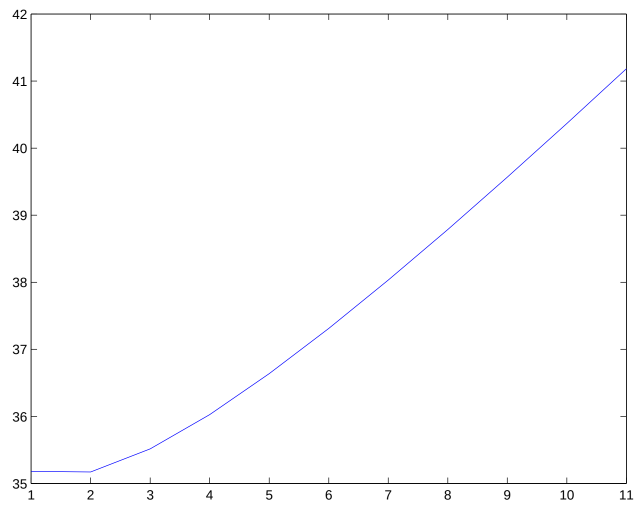

Figure 32: 'Energy-scale' plot of INFINET data in figure 31, which is showing normalcy between the scales 2 and 11.

6.7 Conclusions

This chapter highlights the applicability of wavelets-based network flow analysis techniques for NIDS. The proposed algorithm and method are applicable to detect DoS attacks, wherein data show some bursty behavior. The chapter also indicates the cases where the assumption about the straight line behavior of "energy-scale" plot may be erroneous. The chapter adds two algorithms to the area- first it gives a alternate method to calculate Hurst parameter H and then a hybrid algorithm to get more insight into the type of anomaly. In the later work, integrating the multiscale and SVD techniques, we have proposed an anomaly detection method for self-similar data. The main features of the algorithm are its simplicity and ability in detecting the location and type of anomaly present in data. The implementation results shown for the first data set indicate that the usual 'energy-scale' plot method and the present method have identified same scales of normalcy of data. While the results of INFINET data have shown that the 'energy-scale' plot idea, as we are motivated for the present work, may not always be accurate. However, we have to carry out extensive implementation on various data sets to justify further the wider applicability of our method.

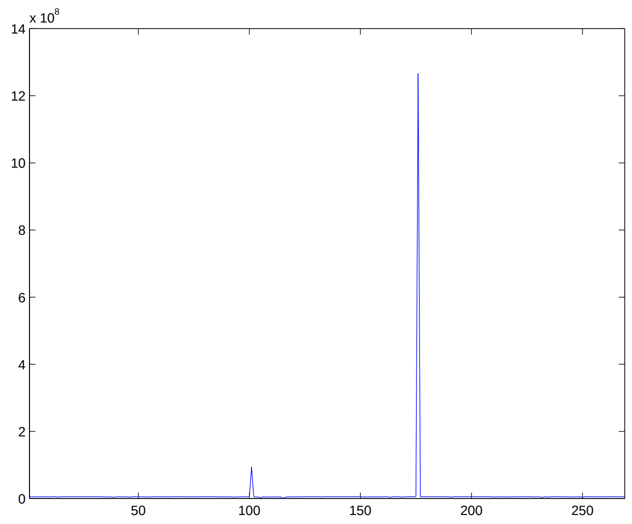

Figure 33: Deviation Scores R_i^d of INFINET data; this figure shows the presence of anomaly in data in figure 31 around 800 and 1400 positions.

The described work in the chapter is published as:
Rawat S, Ch. S. Sastry (2004), "Network Intrusion Detection Using Wavelet Analysis," In: Proc of CIT, 2004, Lecture Notes in Computer Science, Springer, Vol. 3356, pp: 224-232.
Ch. S. Sastry, Rawat S, Pujari A K, Gulati V P (2005), "Network Traffic Analysis using SVD and Multiscale Transforms" Submitted to: Information Sciences, Elsevier.

CHAPTER VII

CONCLUSIONS

To do successful research, you don't need to know everything, you just need to know of one thing that isn't known. -Arthur Schawlow

In this chapter, we provide the summery of the thesis and discuss the possible extensions as future work.

7.1 Summery

The thesis tries to answer the problem, stated earlier as *"How to make the IDS fast enough to process data on-line and detect attacks early and in case of anomaly-based IDS, how to reduce the false positives to an accepted level, with a high detection rate."* Consequently, we propose some techniques, which individually, constitute the answer of the problem.

We study the existing techniques and algorithms to understand their strength and weakness and to get motivation. As a result, we provide a detailed survey on the IDS techniques and methods. In this thesis, our focus has been on anomaly-based IDS. We propose techniques for host- and network-based IDS. For host-based systems, we analyze the system calls, invoked by the processes and for network-based systems, we analyze network packets.

we study the behavior of unix processes in terms of system calls and observe that not all of the system calls, invoked by the processes, are necessary to define its behavior as normal or abnormal. Considering such system calls as redundant or noise, we make use of a linear algebraic techniques, called *singular value decomposition* to reduce the noise. The idea is inspired by an information retrieval technique - *latent semantic indexing*. Though the SVD reduces the dimension of the data by projecting it to some space of lower dimension, it is very difficult to interpret the new dimensions. In order to show the appropriateness of the idea of using SVD, we produce empirical results. We show that SVD removes only the not-so-important system calls, like `mmap`, from the data. We also compare our results with already established scheme to show that reduction in data does not lead to a degradation in accuracy. We show results in terms of ROC curve and AUC score. Such methods are useful in making IDS fast by reducing the data, to be analysed.

Motivated by the work, described in [120], we study the kNN based classifier with cosine metric and find the cases where kNN based scheme may produce some erroneous results. We, therefore, propose a new similarity measure, termed as *binary weighted cosine* (BWC) metric. BWC is based on the frequency and number of common system calls between two processes. BWC metric is used to calculate the similarity and kNN is used to classify the process as normal or abnormal. This scheme represents an example of *distance weighted* kNN classifier. We also think of applying SVD approach further to reduce the data, but as our scheme involves two matrices, it is not feasible to do so. We extend the above scheme by including the partial information about the order of occurrence of individual system calls in the process. For this purpose, we make use of *Kendall Tau distance*, which is used in rank aggregation [51]. We provide experimental results for each of the schemes on DARPA'98 data, as ROC curve and AUC score. The results are compared with relevant scheme, proposed in the literature. However, the calculating the Kendall Tau distance is computationally very expensive. Also, we observe that on the expense of more computational time, the rise in the accuracy is not in that proportion. This point sets the ground and motivation for our next work.

In order to capture the ordering information of system calls, we should use the small sequences of system calls, as they are appearing in the process. We also notice that not whole of the process is abnormal, as compared to normal process. Only a small part (or parts) of an abnormal process is abnormal, while most of it is similar to normal process. We, therefore, project an IDS as a decision table and apply *rough set* [151] based techniques to extract the feature for normal and abnormal processes. The processes corresponding to some attacks and normal ones are represented in a decision table. The *lower approximation* is calculated to discard the overlapping (common) subsequences between normal and abnormal processes. In this way, we get the *positive region* of the given data. We apply a rough set based rule learning algorithm - LEM2 [72] to generate IF-THEN type rules. These rules are used to classify the processes as normal or abnormal. In this way, we are able to analyze the process while it is in running state, thereby making it suitable for on-line intrusion detection system. We experiment on DARPA'98 data, using RSES tool. We are unable to provide AUC score for this scheme as in this case, classification of processes does not produce any ranking. Also, we faced problem while working with RSES, as only GUI is available for use. Therefore, there is no flexibility of customizing it as per the requirement of experiments.

Although, through out our work, we concentrated on host-based IDS, the multiscaling property of wavelets in analyzing network traffic motivated us to explore its usability in IDS. The idea is inspired by the work described in [66][13]. The self-similarity, which is exhibited in network traffic, is taken as the characteristics of normal traffic. The self-similarity is characterized by estimating Hurst parameter H. The loss of self-similarity in the network data can be attributed to the presence of some anomaly. We extend the *energy-scale* plot based scheme for estimating H, by enabling it to detect the locality of the anomaly and the scale on which the anomaly is exhibited. For this purpose, we use wavelet theory and definition of self-similarity. The proposed scheme performs well on KDDcup'99 data. We further provide the extension of the above mentioned scheme. The algorithm proposed herein integrates the wavelet transform with singular value decomposition (SVD) for the analysis of self-similar network traffic data. The algorithm makes use of the properties of the SVD of a matrix of local energies of wavelet coefficients, to determine the scales over which the data have possibly normal behaviour and locations at which the data have possible anomalous behaviour. We concentrate more on the theoretical aspects of our work. To show applicability of our method, we have taken a very small known self-similar data. However, to justify our approach empirically, we apply it on real network data, captured from an operational financial network INFINET [84] and kdd data set [89].

We next discuss some of the possible future extensions of the work summarized above.

7.2 *Future Work*

In this thesis, we present various techniques that, in one or other way, contribute to make IDS more efficient. There are several interesting future directions, out of which few are mentioned below.

- We used SVD to reduce the dimension of the data. But it is difficult to interpret the result. We can use other techniques that reduce the dimension by discarding the features that are not much discriminatory i.e. which system calls are really important to understand the normal behavior of process. In this context, entropy based approaches like *information gain* and rough based techniques like *reduct* can also be used. One good thing about such approaches is that they can calculate explicitly the importance of each system call, and therefore, are easy to interpret the results.

- Almost all of the process behavior-based anomaly approaches are proposed for Unix based system. It should be interesting to apply such approaches on Windows based OS.

- In BWC metric, we consider the frequency of individual system call. Instead of taking single system call, a combination of two or more system calls can be taken. In this way, we can capture co-occurrence of system calls as well, which may produce better results.

- There is a lack of formal analysis methods for IDS. This requires a mathematical model and reasoning based on that. In this direction, one possibility is to consider a process as POMSET P (partially order multiset), by defining a relation '$<$ ' as $s_i < s_j,\ i \neq j \Rightarrow$ system call s_i is followed by s_j, where $s_i, s_j \in P$. This is just an idea and requires further investigation.

- We used wavelets to analyze network data. Wavelets can be also be used on system calls data. Once we construct the incidence matrix A, defined in section 3.3, we can apply wavelet transformation to get each system call's *spectrum* in *frequency-time* domain. This should provide some insight into the way a system call appears, along with other system calls, in a process. This may be useful in monitoring and profiling a process.

7.3 *Concluding Remarks*

During the years, spent on the work reported in this thesis, I experienced moments of joy and sorrow, excitement and resentment. Each of the failures forced me to think more and work hard, and of course, knock my supervisors more frequently. Though it is the end of this thesis, but I find it is the beginning of the journey as a researcher to contribute to and serve the society more.

''The woods are lovely, dark and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep.''
From "Stopping by Woods on a Snowy Evening" - Robert Frost (1875 - 1963)

* * * * *

APPENDIX A

WAVELET THEORY

The key idea in wavelet technique is to start with a function ϕ that is made up of a smaller version of itself, i.e.,

$$\phi(x) = \sum_{k=-\infty}^{\infty} h_k \phi(2x - k), \tag{43}$$

called, the refinement (or 2-scale) equation. The coefficients h_k are called filter coefficients or masks. The function ϕ is called scaling (or father wavelet) function.

To compute the wavelet transform of a function efficiently, the concept of Multi Resolution Analysis (MRA) was introduced [40]. There is a family of algorithms, that can be computed fast, associated with MRA. The motivation of MRA is to use a sequence of embedded subspaces to approximate $L^2(I\!R)$, the space of all finite energy functions defined over the real line $I\!R = (-\infty, \infty)$, so that one can choose a proper subspace for a specific application task to get a balance between accuracy and efficiency (bigger spaces can contribute better accuracy at the expense of computational resources). Mathematically, MRA concerns with the property of a sequence of closed subspaces $V_j, j \in Z$ which approximate $L^2(I\!R)$ and satisfy

$$\{0\} = V_{-\infty} \subset \ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots \subset V_\infty = L^2(I\!R), \tag{44}$$

which confirms that the closure of union of all V_j is $L^2(I\!R)$. The multiresolution property is reflected by the property: $f(.) \in V_j \iff f(2.) \in V_{j+1}$, meaning that all the spaces are scaled versions of the central space V_0. Firstly, the translates of ϕ, i.e., $\{\phi(x-k)\}_{k \in Z}$, generate an orthogonal basis for V_0. The nestedness property of V_j spaces ensures that the functions $\{\phi(2^j x - k)\}_{k \in Z}$ generate an orthogonal basis for V_j for each j. It may be noted that the translation parameter k controls the observation location, while the scaling parameter j controls the observation resolution.

Having an orthonormal basis of V_j is only half of the picture. In order to solve the problems such as noise filtering, we need to have a way of isolating the "spikes" that belong to V_j, but that are not members of V_{j-1}. This is where the wavelet ψ enters the picture. Let W_0 be the complement of V_0 in V_1, i.e., $W_0 = V_1 \ominus V_0$ or $V_1 = V_0 \oplus W_0$. The symbols $\ominus$ and $\oplus$ have the following meaning: If every function in V_1 is uniquely written as sum of the orthogonal components of it in V_0 and W_0, (i.e., $h \in V_1$ implies $h = h_1 + h_2$ for unique $h_1 \in V_0$, $h_2 \in W_0$ and $\int_{-\infty}^{\infty} h_1(x)h_2(x)dx = 0$), we write $V_1 = V_0 \oplus W_0$ or $W_0 = V_1 \ominus V_0$. The question as to how the information in W_0 can be studied is answered by the wavelet function ψ, called mother wavelet, which is defined [40] by

$$\psi(x) = \sum_{k=-\infty}^{\infty} (-1)^k \overline{h}_{1-k} \phi(2x - k). \tag{45}$$

From the definition of W_0, it can be concluded that $\psi \in V_1$, but $\psi \notin V_0$. If $\{\psi(x - k)\}_{k \in Z}$ generates an orthonormal basis of W_0, for any j, translates of $\psi(2^j.)$ generate a basis of W_j, which is $V_{j+1} \ominus V_j$. Using the definition of V_j and W_j spaces, for some $J \geq 0$, we have

$$\begin{aligned} V_J &= V_{J-1} \oplus W_{J-1} \\ &= V_{J-2} \oplus W_{J-2} \oplus W_{J-1} \\ &= \ldots \\ &= V_0 \oplus W_0 \oplus \ldots \oplus W_{J-1}. \end{aligned} \tag{46}$$

Decomposing V_0 further, we have

$$V_J = W_{-\infty} \oplus \ldots W_0 \oplus \ldots \oplus W_{J-1}. \tag{47}$$

Letting $J \to \infty$, we get

$$L^2 \leftarrow V_J = W_{-\infty} \oplus \ldots W_0 \oplus \ldots \oplus W_{J-1} \to \oplus_{j=-\infty}^{\infty} W_j. \tag{48}$$

In equation (48), L^2 space is partitioned into different subspaces that are orthogonal to each other. Hence, a function $f \in L^2$ can be divided into different 'orthogonal' pieces g_j, that capture the 'edges' or 'singularities' of f of different 'strengths', as follows

$$f = \sum_{j \in Z} g_j \quad \text{for} \quad g_j \in W_j. \tag{49}$$

In equation (49), $g_j = \sum_{k \in Z} \langle f, \psi_{j,k} \rangle \psi_{j,k}$ and $\langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(x) \psi_{j,k}(x) dx$, the innerproduct of f and $\psi_{j,k}$. $\psi_{j,k}(t) = 2^{\frac{j}{2}} \psi(2^j t - k)$. The factor $2^{\frac{j}{2}}$ in the definition of $\psi_{j,k}$ is taken to normalize the energy of it. In actual computational work, using equation (44), a function $f \in L^2$ is approximated by $f_J \in V_J$ for a fair choice of J. Then, using equation (46), we take the wavelet representation of f as

$$f \approx f_J = f_0 + g_0 + \ldots + g_{J-1}. \tag{50}$$

Here, $f_0 \in V_0$, and hence $f_0 = \sum_k < f, \phi_{0,k} > \phi_{0,k}$, and $g_i \in W_i$. In equation (50), f_0 captures the approximate information or "identity" of the function, while the functions g_i capture the edges or "spikes" present in f of different "strengths", as depicted in figure 35.

A direct application of MRA is the fast discrete wavelet algorithm, called pyramid algorithm [133]. The basic idea in pyramid algorithm is the progressive reconstruction feature, which involves the addition of details to coarser approximation for better representation.

Using equations (43) and (45), the approximation and the detail coefficients at the j^{th} level of resolution, denoted by $c^j = \{< f, \phi_{j,k} >\}$, $d^j = \{< f, \psi_{j,k} >\}$ respectively, are computed via the following iterative decomposition formulae

$$\begin{aligned} c_l^j = \sum_k h_{k-2l} c_k^{j+1} &= \left[\downarrow_2 \left(\overline{h} \star c^{j+1} \right) \right]_l \\ d_l^j = \sum_k g_{k-2l} c_k^{j+1} &= \left[\downarrow_2 \left(\overline{g} \star c^{j+1} \right) \right]_l . \end{aligned} \tag{51}$$

Here, the following notations are used: $h = \{h_n\}$, $g = \{g_n = (-1)^n h_{1-n}\}$ with $\overline{h}_n = h_{-n}$ and $\downarrow_2 \{h_n\} = \{h_{2n}\}$, called the down sampling operation. The *star* or * operation stands for the convolution operation. The wavelet transform that is computed without down sampling operation is called undecimated wavelet transform, which is a redundant and shift invariant transform. It has been observed that the undecimated transform is shown to be useful in such applications as denoising [106]. The decomposition/reconstruction process of a function is shown in figure 34.

Abusing the convention slightly for the sake of convenience, in the later sections, we refer to the finest resolution level L as 0 level, while its subsequent coarse levels as $1, 2, \ldots$ (i.e, when we say 'level j', it actually means $L - j$).

$$\begin{tikzcd}
 & d_{J-1} & d_{J-2} & & & d_{J-p+1} \\
c_J \arrow[r] \arrow[ur] & c_{J-1} \arrow[r] \arrow[ur] & c_{J-2} \cdots\cdot \arrow[rr] & & c_{J-p+1} \arrow[r] \arrow[ur] & c_{J-p}
\end{tikzcd}$$

Decomposition Algorithm

$$\begin{tikzcd}
d_{J-p} \arrow[dr] & d_{J-p+1} \arrow[dr] & & d_{J-1} \arrow[dr] & & \\
c_{J-p} \arrow[r] & c_{J-p+1} \arrow[r] & c_{J-p+2} \cdots\cdot \arrow[rr] & & c_{J-1} \arrow[r] & c_J
\end{tikzcd}$$

Reconstruction Algorithm

Figure 34: Decomposition and reconstruction processes

In the following, we summarize some of the properties of wavelet transform.

Multiresolution frame work: As in the decimal system of representation of numbers, in wavelet decomposition, details are added successively to coarser level information (see equation (50)). This progressive reconstruction feature is critical in many applications including the ones in IDS.

Zero moments: We say that a wavelet system has zero (vanishing) moment of the order N if the following property is implied [40].

$$\int_{-\infty}^{\infty} x^m \psi(x) dx = 0. \tag{52}$$

where $m = 0, \cdots, N-1$. This property reveals the oscillatory nature of ψ, which is an important factor [40] determining the speed of convergence of wavelet series in equation (49), and the sparsity in the wavelet domain of a function etc. Higher order zero moments ensure that data in wavelet domain (under appropriate smoothness conditions on data) can become sparse, i.e., many wavelet coefficients are negligibly small. This feature is useful in data reduction problems.

Computational Complexity: A vector matrix multiplication is an $O(N^2)$ procedure. Hence, the Discrete Fourier transform is an $O(N^2)$ procedure. Which is achieved in $O(N \log_2 N)$ computations by fast Fourier transform. However, the fast wavelet algorithm [133] based

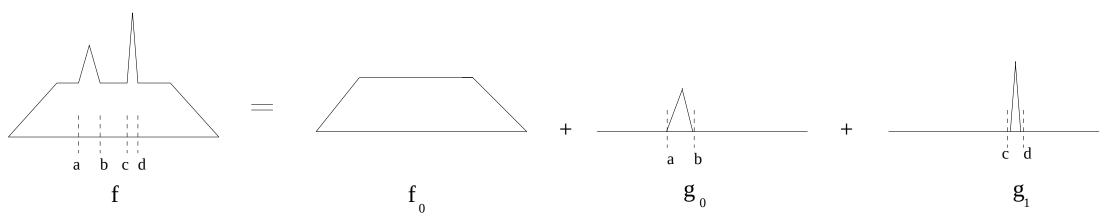

Figure 35: Multiresolution decomposition

on the pyramidal algorithm achieves the same in $O(N)$ computations [18]. Consequently, the complexity involved in the wavelet computations varies linearly with the size of data (N).

Publications from the Thesis

I. Sanjay Rawat, V. P. Gulati, Arun K. Pujari "Frequency And Ordering Based Similarity Measure For Host Based Intrusion Detection". Journal of Information Management and Computer Security, 12(5). Emerald Press. 2004. pp. 411-421

II. Sanjay Rawat, Arun K. Pujari, V. P. Gulati "On the Use of Singular Value Decomposition for a Fast Intrusion Detection System". In: Proc. of the First International Workshop on Views On Designing Complex Architectures (VODCA 2004), Bertinoro, Italy, ENTCS, Elsevier.

III. Sanjay Rawat, Challa S. Sastry "Network Intrusion Detection System Using Wavelet Analysis". In: Proc of the 7th International Conference on Information Technology (CIT 2004), Hyderbad, India. LNCS # 3356, Springer. pp 224-232

IV. Sanjay Rawat, V. P. Gulati, Arun K. Pujari "An Intrusion Detection Scheme Based on Frequency and Ordering". In: Proc of the National Workshop on Cryptology - 2003, Chennai, India. 2003

V. Sanjay Rawat, Arun K. Pujari, V. P. Gulati "A Fast Host-Based Intrusion Detection System Using Rough Set Theory" To appear in: Transactions on Rough Sets, Vol 4, Springer.

VI. Sanjay Rawat , Arun K. Pujari, V. P. Gulati, V. Rao Vemuri (2004)"Intrusion Detection using Text Processing Techniques with a Binary-Weighted Cosine Metric" Submitted to: International Journal of Information Security, Springer.

VII. Challa S Sastry, Sanjay Rawat , Arun K. Pujari, V. P. Gulati (2005) "Network Traffic Analysis using SVD and Multiscale Transforms" Submitted to: Information Sciences, Elsevier.

VIII. Sanjay Rawat , Arun K. Pujari, V. P. Gulati (2005) "Intrusion Detection Systems: A Survey on Algorithms and Methods." Technical Report. Submitted to Journal.

REFERENCES

[1] Abramovich F, Bailey T, Sapatinas T (2000), "Wavelet Analysis and its Statistical Applications," *JRSSD*,48:1-30

[2] Abry P, Veitch D (1998), "Wavelet Analysis of Long-Range Dependent Traffic," *IEEE Trans. Inform. Theory* 44:2-15

[3] Abry P, Flandrin P, Taqqu M S, Veitch D (2003), "Self-Similarity and Long-Range Dependence Through the Wavelet Lens," *Theory and Applications of Long-Range Dependence*, P. Doukhan, G. Oppenheim, and M. S. Taqqu (Editors), Birkhauser, pp 526-556

[4] Aicklin U, Bentley P, cayzer S, Kim J, McLeod J (2003), "Danger Theory: The Link Between AIS and IDS". In: Proc of the Second International Conference on Artifiical immune Systems (ICARIS-03). pp 147-155

[5] Akkus A, Güvenir H A (1996), "Nearest Neighbor Classification on Feature Projections". In: Proceedings of the 13 International Conference on Machine Learning, Lorenza Saitta (Ed.),Bari, Italy. Morgan Kaufmann. pp 12-19

[6] Allen W H and Marin G A (2003), "On the Self-Similarity of Synthetic Traffic for the Evaluation of intrusion Detection Systems," *Proc. of the IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, Orlando, FL. pp 242-248

[7] Allen W H and Marin G A (2004), "The Loss Technique for Detecting New Denial of Service Attacks," *Proc IEEE South East Conference*, Greensboro, NC

[8] Anderson James P (1980), "Computer Security Threat Monitoring and Surveillance". Technicap Report Contract 79F26400, James P Anderson Co., Box 42, Fort Washington, PA, 19034, USA.

[9] Axelsson S (1999), "Research in Intrusion Detection Systems: A Survey". Technical Report No. 98-17, Dept. of Computer Engineering, Chalmers University of Technology, Gteborg, Sweden.

[10] Bace R, Mell P (2001), "NIST Special Publication on Intrusion Detection System". SP800-31, NIST, Gaithersburg, MD.

[11] Barbara D, Couto J, Jajodia S, Wu N (2001a), "ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection". SIGMOD Record, 30(4): 15-24

[12] Barbara D, Couto J, Jajodia S, Wu N (2001b), "ADAM: Detecting Intrusions by Data Mining". In: Proc of the IEEE SMC Information Assurance Workshop, West Point, NY. pp 11-16

[13] Barford P and Plonka D (2001), "Characteristics of Network Traffic Flow Anomalies," *Proceedings of ACM SIGCOMM Internet Measurement Workshop IMW*.

[14] Barford P, Kline J, Plonka D (2002), "A Signal Analysis of Network Traffic Anomalies". In: Proc of ACM SIGCOMM Internet Measurement Workshop.

[15] Bass T (2000), "Intrusion detection systems and multisensor data fusion". Communications of the ACM. 43(4):99-105

[16] Beran J (1994), *Statistics for Long-Memory Processes*, Chapman and Hall, New York.

[17] Berry M W, Dumais S T, O'Brien G W (1995), "Using Linear Algebra for Intelligent Information Retrieval". SIAM Review vol. 37(4):573-595

[18] Beylkin G, Coifman R, Rokhlin V (1991), "Fast Wavelet Transforms and Numeraical Algorithms". Comm. Pure and Appl. Math, 44:141-183

[19] Biermann E, Cloete E, Venter L M (2001), "A Comparison of Intrusion Detection Systems". Computers & Security 20(8): 676-683

[20] Bivens A, Palagiri C, Smith R and Szymanski B (2002), "Network-Based Intrusion Detection Using Neural Network". In: Proc of Artificial Neural Networks in Engineering (ANNIE'02)

[21] Botha M, Solms R von (2003), "Utilizing Fuzzy Logic and Trend Analysis for Effective Intrusion Detection". Computers & Security 22(5):423-434

[22] Breunig M M, Kriegel H -P, Ng R T, Sander J (2000), "LOF: Identifying Density-Based Local Outliers". In: Proc of the ACN SIGMOD conference.

[23] Cabrera J, Ravichandran B and Mehra R (2000), "Statistical Traffic Modeling for Network Intrusion Detection". In: Proc of the 8th IEEE Symposium on Modeling, Analysis and simulation of Computers and Telecommunications, San Francisco, California. pp 466-475

[24] Cabrera J B D, Ravichandran B, Mehra R K (2001), "Detection and Classification of Intrusions and Faults Using Sequences of System Calls". In: ACM SIGMOD Record, Special Issue: Special Section on Data Mining for Intrusion Detection and treat Analysis, Vol. 30(4). pp 25-34

[25] Cai Z, Guan X, Shao P, Peng Q, Sun G (2003), "A Rough Set Theory Based Method for Anomaly intrusion Detection in Computer Network Systems". J Expert System 20(5):251-259

[26] Cannady J (2000a), "Aplying CMAC-Based On-Line Learning to Intrusion Detection". In: Proc of the 2000 IEEE/INNS International Joint Conference on Neural Networks (IJCNN'00), Como, Italy. pp. 405-410

[27] Cannady J (2000b), "Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks". In: Proc of the 23^rd National Information System Security Conference, Baltimore, Maryland.

[28] Chan Z, Zhu B (2000), "Some Formal Analysis of the Rocchio's Similarity-based Relevance Feedback Algorithm". Technical Report CS-00-22, Dept. of Computer Science, University of Texas-Pan American, Edinburg, TX.

[29] Chawla N V, Bowyer K Lazarevic A, Hall L O, Kegelmeyer W P (2002), "SMOTE: Synthetic Minority Over-Sampling Technique". Journal of Artificial Intelligence Research, Vol. 16:321-357

[30] Chawla N V, Lazarevic A, Hall L O, Bowyer K (2002), "SMOTEBoost: Improving Prediction of the Minority Class in Boosting". AHPCRC Technical Report. University of Minnesota.

[31] Cios Krzysztof, Pedrycz Witold, Swiniarski Roman W (2000), *Data mining methods for Knowledge discovery*. Kluwer Academic Publisher USA.

[32] Cohen W W (1995), "Fast Effective Rule Induction". In Machine Learning: Proc of the 12th International Conference, California. Morgan Kaufmann.

[33] Crovella M and Bestavros A (1997), "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE-ACM Transactions on Networking*, 5:835-846

[34] Crosbie M, Spafford E H (1995), "Applying Genetic Programming to Intrusion Detection". In: Working Notes for the AAAI Symposium on Genetic Programming. MIT Cambridge, MA, USA. pp 1-8

[35] DARPA 1998 Data, MIT Lincoln Laboratory.
http://www.ll.mit.edu/IST/ideval/data/data_index.html

[36] Dasgupta D (1999), "Immunity-Based Intrusion Detection Systems: A General Framework". In: Proc of the 22nd National Information Systems Security Conference (NISSC).

[37] Dasgupta D, Gonzalez F (2002), "An Immunity-Based Technique to Characterize Intrusions in Computer Networks". IEEE Transactions on Evolutionary Computation vol. 6(3):281-291

[38] Debar H, Dacier M, Nassehi M, Wespi A (1998), "Fixed vs. Variable-Length Patterns for Detecting Suspicious Process Behavior". In: Proc of the 5th European Symposium on research in computer security ESORICS 1998, Belgium. pp 1-15

[39] Dao Vu N P and Vemuri R (2002), "A Performance Comparison of Different Back Propogation Neural Networks Methods in Computer Network Intrusion Detection". J. of Differential Equations and Dymanic Systems.

[40] Daubechies I (1992), *Ten lectures on wavelets*. CBMS-NSF Series in Appl. Math., 61, SIAM Philadelphia

[41] Deerwester S, Dumais, Susan T, Furnas, G W, Landauer T K, Harshman R (1990), "Indexing by Latent Semantic Analysis". Journal of the American Society of Information Science, Vol. 41(6):391-407

[42] Denning D E (1990), "An Intrusion-Detection Model". In: Proceedings of the 1986 IEEE Symposium on Security and Privacy (SSP '86), IEEE Computer Society Press. pp. 118-133

[43] Denoeux T (1995), "A k-Nearest Neighbor Classification Rule Based on Dempster-Shafer Theory. IEEE Transactions on Systems, Man and Cybernetics. vol. 25:804 813

[44] D'haeseleer P, Forrest S, Helman P (1996), "An immunological approach to change detection: algorithms, analysis, and implications". In: Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press. pp. 110-119

[45] D'haeseleer P, Forrest S, Helman P (1997), "A Distributed Approach to Anomaly Detection". Draft, University of New Maxico. Available at: http://www.cs.unm.edu/$\tilde{}$forrest/isa_papers.htm

[46] Dickerson J E, Dickerson J A (2000), "Fuzzy Network Profiling for Intrusion Detection." In: Proc of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, Atlanta, July. pp 301-306

[47] Dickerson J E, Juslin J, Koukousoula O, Dickerson J A (2001), "Fuzzy Intrusion Detection". In: IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference, Volume 3. Vancouver, British Columbia. pp 1506-1510

[48] Dokas P, Ertoz L, Kumar V, Lazarevic A, Srivastava J, Tan P (2003), "Data Mining for Network Intrusion Detection". In: Proc of the NSF Workshop on Next Generation Data Mining, Baltimore, MD.

[49] Pong Chan K, Fu A W C (1999), "Efficient Time Series Matching by Wavelets". In: Proc of ICDE. pp 126-133

[50] Dudani S A, (1976), "The Distance-Weighted k-Nearest-Neighbor Rule". IEEE Trans. Syst. Man Cyber. vol. 6:325327

[51] Dwork C, Kumar R, Naor M, Sivakumar D (2001), "Rank Aggregation Methods for the Web". In: Proceedings of the tenth International World Wide Web Conference-2001. pp 613-622

[52] Ertoz L, Eilertson E, Lazarevic A, Tan P, Srivastava J, Kumar, V, Dokas P (2003), "The MINDS - Minnesota Intrusion Detection System". Accepted for the book *Next Generation Data Mining*.

[53] Eskin E (2000), "Anomaly Detection Over Noisy Data Using Learned Probability Distribution". In: Proc of the 2000 International conference on Machine Learning (ICML'2000). Palo Alto, CA.

[54] Eskin E, Miller M, Zhong Zhi-Da, Zhang J, Stolfo S (2000), "Adaptive Model Generation for Intrusion Detection Systems". In: Proc of the ACMCCS Workshop on Intrusion Detection and Prevention. Greec.

[55] Eskin E, Arnold A, Prerau M, Portnoy L, Stolfo S (2002), "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data". In: Barbara D, Jajodia S (eds.) *Applications of Data Mining in Computer Security*. Kluwer academics Publishers, pp 77-102

[56] Fan W, Miller M, Stolfo S J, Lee W, Chan P K (2001), "Using Artificial Anomalies to Detect Unknown and Known Network Intrusions". In: Proc of the IEEE International Conference on Data Mining ICDM 200, San Jose, California, USA. pp 123-130

[57] Feldmann A, Gilbert A, Willnger W, Kurtz T (1997), "Looking Behind and Beyond Self-Similarity: Scaling Phenomena in Measured WAN Traffic". In: Proc. of 35th Annual Allerton Conference on Communication, Control, and Computing. pp 269-280

[58] Feldmann A., Gilbert A., Willnger W. and Kurtz T. (1997), "The Changing Nature of Network Trafiic:Scalling Phenomena," ACM SIGCOMM Computer Communication Review, 28(2):5-29

[59] Forrest S, Hofmeyr S A, Somayaji A (1997), "Computer Immunology". Communications of the ACM, 40(10):88-96

[60] Forrest S, Hofmeyr S A, Somayaji A, Longstaff T A (1996), "A Sense of Self for Unix Processes". In: Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy. Los Alamitos, CA. IEEE Computer Society Press. pp 120-128

[61] Frank J (1994), "Artificial Intelligence and Intrusion Detection: Current and Future Directions". In: Proc of the National 17th Computer Security Conference.

[62] Freund Y, Schapire R (1996), "Experiments with a New Boosting Algorithm". In: Proceedings of the 13th International Conference on Machine Learning. pp 325-332

[63] Garvey T, Lunt T F (1991), "Model-based Intrusion Detection". In: Proceedings of the 14th National Computer Security Conference. pp 372-385

[64] Ghosh A K, Schwartzbard A (1999a), "A Study in Using Neural Networks for Anomaly and Misuse Detection". In: Proceedings of the 8th USENIX security Symposium. Aug. 23-26, Washington D C USA, pp 141-151

[65] Ghosh A K, Schwartzbard A, Schatz M (1999b), "Learning Program Behavior Profiles for Intrusion Detection". In: Proc of the 1st Workshop on Intrusion Detection and Network Monitoring, USENIX. April 9-12, 1999, Santa Clara, California, USA. pp 51-62

[66] Gilbert A C (2001), "Multiscale Analysis and Data Networks". Applied and Computational Harmonic Analysis 10:185-202

[67] Golub G H, van Loan C F (1996), *Matrix Computations*. John Hopkins University Press, 3rd edition.

[68] Gmez J, Gonzlez F, Dagupta D (2003), "An Immuno-Fuzzy Approach to Anomaly Detection". In: Proc of The IEEE International Conference on Fuzzy Systems (FUZZIEEE), St. Louis, MO. pp 1219-1224

[69] Gonzalez L J (2002), " Current Approaches to Detecting Intrusions". Assignment for Doctoral Course, DCIS 1200 Directed Independent Study, Nova Southeastern University, Computer Information Systems.

[70] Gu J, Lee D, Park S, Sim K (2000), "An Immunity-Based Security Layer Model". In: Proc of the Genetic and Evolutionary Computation Conference, Workshop on Artificial Immune Systems and their Applications. pp 47-48

[71] Guven A, Sogukpinar I (2003), "Understanding User's Keystroke Patterns for Computer Access Security". Computers & Security, 22(8):695-706

[72] Grzymala-Busse J W (1997), “A New Version of the Rule Induction System LERS”. Fundamenta Informaticae, 31(1):27-39

[73] Han Sang-J, Cho Sung-B (2003), “Detecting Intrusion with Rule-Based Integration of Multiple Models”. Computers & Security, 22(7):613-623

[74] Hand D J, Till R J(2001), “A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems”. Machine Learning 45(2):171-186

[75] Helman P, Liepins G (1993), “Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse”. IEEE Transactions on Software Engineering, 19(9): 886-901

[76] Hershkop S, Ferster R, Bui L H, Wang K, Stolfo S J (2003), “Host-based Anomaly Detection by Wrapping File System Accesses”. CU Tech Report, in submission.

[77] Hofmeyr S A, Forrest A, Somayaji A (1998), “Intrusion Detection Using Sequences of System Calls”. Journal of Computer Security 6:151-180

[78] Hofmeyr S (1999), “An Immunological Model of Distributed Detection and its Application to Computer Security”. PhD Thesis, University of New Maxico.

[79] Hu Wenjie, Liao Y, Vemuri V (2003), “Robust Support Vector Machines for Anamoly Detection in Computer Security”. In: International Conference on Machine Learning, Los Angeles, CA.

[80] Huang P, Feldmann A, Willinger W (2001), “A Non-Intrusive, Wavelet-Basesd Approach to Detect Network Performance Problems”. In: Proc of the First ACM SIGCOMM Workshop on Internet Measurement IMW’01, San Francisco, California, USA. pp 213-227

[81] Hyum Oh S, Lee W S (2003), “An Anomaly Intrusion Detection Method by Clustering Normal User Behavior”. Computers & Security, 22(7):596-612

[82] Ilgun K (1993), “USTAT: A Real-Time Intrusion Detection System for UNIX”. In: Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy. pp. 16-28

[83] Ilgun K, Kemmerer R A, Porras P A (1995), “State Transition Analysis: A Rule-Based Intrusion Detection Approach”. IEEE Transactions on Software Engineering 21(3): 181-199

[84] Indian Financial Network (InFiNet). URL:
http://www.idrbt.ac.in/infinet/infinet.html

[85] Javitz H S, Valdes A (1994), “The NIDES Statistical Component Description and Justification”. Technical Report A010. SRI.

[86] Jiang N, Hua K A, Oh JungHwan (2003), “Exploiting Pattern Relationship for Intrusion Detection”. In: Proc of the 2003 Symposium on Applications and the Internet (SAINT’03). IEEE Computer Society.

[87] Jiang N, Hua K A, Sheu S (2002), “Considering Both Intra-Pattern and Inter-Pattern Anomalies for Intrusion Detection”. In: Proc of the ICDM 2002, Maebashi City, Japan. pp 637-640

[88] Julisch K (2002), “Data mining for intrusion detection: A critical review”. In: Barbara D, Jajodia S (eds.) *Applications of Data Mining in Computer Security*, Kluwer academics Publishers,Boston.

[89] KDD 1999 data set, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[90] Kelly J D, Davis L (1991), “Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm”. In: Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications (ICGA). pp 377-383

[91] Kemmerer R A (1998), “NSTAT: A Model-based Real-time Network Intrusion Detection System”.Technical Report, Number TRCS97-18, Computer Science, University of California, Santa Barbara.

[92] Kephart J O (1994), “A Biologically Inspired Immune System for Computers”. In: R.A. Brooks & P. Maes (eds.), Artificial Life IV Proc of the Fourth International workshop on the Synthesis and Simulation of Living Systems, MIT Press. pp 130-139

[93] Kim J, Bentley P (1999a), “The Artificial Immune Model for Network Intrusion Detection”. In: Proc of 7th European Conference on Intelligent Techniques and Soft Computing (EUFIT’99), Aachen, Germany.

[94] Kim J, Bentley P (1999b), “The Human Immune System and Network Intrusion Detection”. In: Proc of the 7th European Conference on Intelligent Techniques and Soft Computing (EUFIT ’99), Aachen, Germany.

[95] Kim J, Bentley P (2001a), “Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection”. In: Proc of GECCO’2001. pp 1330-1337

[96] Kim J, Bentley P (2001b), “Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. In the Congress on Evolutionary Comoutation (CEC-2001), Seol, Korea. pp 1244-1252

[97] Kosoresow A P, Hofmeyr S A (1997), “Intrusion Detection via System Call Traces”. IEEE Software, 14(5):35-42

[98] Kruegel C, Mutz D, Valeur F, Vigna G (2003a), “On the Detection of Anomalous System Call Arguments”. In: Proc of the 8th European Symposium on Research in Computer Security (ESORICS), Norway, October 13-15, 2003, LNCS# 2808. pp 326-343

[99] Kruegel C, Toth T (2003b), “Using Decision Trees to Improve Signature-based Intrusion Detection”. In: Proc of the Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003, Pittsburgh, PA, USA. LNCS# 2820 Springer 2003. pp 173-191

[100] Kruegel C, Toth T, Kirda E (2002), “Service Specific Anomaly Detection for Network Intrusion Detection”. In: Symposium on Applied Computing (SAC), ACM Digital Libray, Spain.

[101] Kumar S, Spafford E (1994), “A pattern-matching model for intrusion detection”. In: Proceedings National Computer Security Conference. pp. 11-21

[102] Labib K, Vemuri R (2002), "NSOM: A Real-Time Network-Based Intrusion Detection System Using Self-Organizing Maps". Submitted to Network & Security.

[103] Lamont G B, Marmelstein R E and Van Veldhuizen D A (1999), "A Distributed Architecture for a Self-Adaptive Computer Virus Immune System". D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*.McGraw hill, London. pp 167-183

[104] Lane T, Brodley C E (1999), "Temporal Sequence Learning and Data Reduction for Anomaly Detection". ACM Transections Information System Security, 2(3): 295-331

[105] Lane T, Brodly C E (1997), "An application of Machine Learning to Anomaly Detection". In: Proceeding of the 20th National Information System Security Conference. pp366-377

[106] Lang M, Guo H, Odegard J E, Burrus C S (1996), "Noise reduction using an undecimated discrete wavelet transform," IEEE Signal Processing Letters, 3(1):10-12

[107] Lazarevic A, Dokas P, Ertoz L, Kumar V, Srivastava J, Tan P (2002), "Cyber Threat Analysis - A Key Enabling Technology for the Objective Force (A Case Study in Network Intrusion Detection)". In:Proc of the 23rd Army Science Conference, Orlando, FL.

[108] Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J (2003), "Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection". In: Proc of the 3rd SIAM Conference on Data Mining, San Francisco, CA.

[109] Leach J (2003), "Improving User Security Behavior". Computers & Security, 22(8):685-692

[110] Lee W (2002), "Applying Data Minig to Intrusion Detection: The Quest for Automation, Efficiency and Credibility". ACM SIGKDD Explorations, 4(2):35-42

[111] Lee W, Stolfo S, Chan P, Eskin E, Fan W, Miller M, Hershkop S, Zhang J (2001a), "Real Time Data Mining-based Intrusion Detection". In: Proc of The 2001 DARPA Information Survivability Conference and Exposition (DISCEX II), Anaheim, CA.

[112] Lee W, Xiang Dong (2001b), "Information-Theoretic Measures for Anomaly Detection". In: Proc of the IEEE Symposium on Security and Privacy,Oakland, California, USA. pp 130-143

[113] Lee W, Stolfo S, Mok K (1999a), "Mining in a Data-flow Environment: Experience in Network Intrusion Detection". In: Proc of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, CA. pp 114-124

[114] Lee W, Stolfo S, Mok K W (1999b), "A Data Mining Framework for Building Intrusion Detection Models". In: Proc of the 1999 IEEE Symposium on Security and Privacy, Oakland, CA. pp 120-132

[115] Lee W, Stolfo S, Mok K W (1998a), "Mining Audit Data to Build Intrusion Detection Models". In: Proc of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98), New York, NY.

[116] Lee W, Stolfo Salvatore J (1998b), “Data Mining Approaches for Intrusion Detection”. In: Proceedings of the 7th USENIX Security Symposium (SECURITY-98), Usenix Association, January 26-29.pp. 79-94

[117] Lee W, Stolfo S, Chan P (1997), “Learning Patterns from Unix Process Execution Traces for Intrusion Detection”. In: Proceedings of the AAAI97 workshop on AI Methods in Fraud and Risk Management. AAAI Press. pp 50-56

[118] Leland W, Taqqu M S, Willinger W, Wilson D V (1994), “On The Self-Similar Nature of Ethernet Traffic (extended version)”. IEEE/ACM Transactions on Networking, 2:1-15

[119] Li T, Li Q, Zhu S Ogihara M (2002), “A Survey of Wavelet Applications in Data Mining”. ACM SIGKDD Explorations, 4(2):49-68

[120] Liao Y, Vemuri V R (2002a), “Use of K-Nearest Neighbor Classifier for Intrusion Detection”. Computers & Security, 21(5):439-448

[121] Liao Y, Vemuri V R (2002b), “Using Text Categorization Techniques for Intrusion Detection”. In: Proceedings USENIX Security 2002. San Francisco. pp 51-59

[122] Lichodzijewski P, Nur Zincir-Heywood A and Heywood M I (2002), “Host-Based Intrusion Detection Using Self-Organizing Maps”. In: Proc of International Joint Conference on Neural Networks (IJCNN’02). pp 1720-1725

[123] Lichodzijewski P, Nur Zincir-Heywood A, Heywood M I (2002), “Dynamic Intrusion Detection Using Self-Organizing Maps”. In: Proc of the 14^th Annual Canadian Information Technology security Symposium (CITSS’02)

[124] Lin T Y (1994), “Anomaly Detection- A Soft Computing Approach”. In: Proc of the 1994 Workshop on Security Paradigms. IEEE Computer Society Press.

[125] Lippman R P, Fried D J, Graff I, Haines J W, Kendell K R, McClung D, Weber D, Webster S E, Wyshofrod D, Cunningham R K, Zissman M A (2000), “Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation”. In: DISCEX’00 -DARPA Information Survivability Conference & Exposition, Hilton Head, SC, vol 2. pp. 12-26

[126] Lippmann R P, Cunningham R K (1999), “Improving Intrusion Detection Performance Using Keyword Selection and Neural Netwprks”. In: Proc of RAID’99. Indiana, USA.

[127] Lunt T F (1990), “Using Statistics to Track Intruders”. In: Proceedings of the Joint Statistical Meetings of the American Statistical Association.

[128] Lunt T F, Tamaru A, Gilham F, Jagannathan R, Neumann P G, Javitz H S, Valdes A, Garvey T D (1992), “A Real-Time Intrusion Detection Expert System (IDES)”. Technical Report, SRI Computer Science Laboratory.

[129] Lunt T (1993), “Detecting Intruders in Computer Systems”. In: Proc of the 1993 Coference on Auditing and Computer Technology.

[130] Mahoney M V, Chan P K (2002), “Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks”. In: Proc of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. Edmonton, Alberta, Canada. pp 376 - 385

[131] Mahoney M V, Chan P K (2003), “Learning Rules for Anomaly Detection of Hostile Network Traffic”. In: Proc of the Third International Conference on Data Mining (ICDM 2003), Melbourne FL.IEEE.4 pages.

[132] Mahoney M V, Chan P K (2003), “Learning Models of Network Traffic for Detecting Novel Attacks”. In: Proc of 8^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada. pp 377-385

[133] Mallat S (1998), *A Wavelet Tour of Signal Processing*. Academic Press.

[134] Mandelbrot B B (1969), “Some Long-run Properties of Geophysical Records”. Water Resources Research, 5:321340

[135] Manganaris S, Chritensen M, Zerkle D, Hermiz K (2000), “A Data Mining Analysis of RTID Alarms”. Computer Networks, 34:571-577

[136] Matzinger P (1998), “An Innate sense of Danger”. Seminars in Immunology, 10:399-415

[137] MATLAB -The Language of Technical Computing. http://www.mathworks.com

[138] Maxion R A, Tan M C (2000), “Benchmarking Anomaly-Based Detection Systems”. In: Proc of the 1st international Conf on Dependable Systems and Networks. New York, USA. pp 623-630

[139] 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):262 294

[140] Meadows C (2001), “A Cost-Based Framework for Analysis of Denial of Service in Networks”. Journal of Computer Security, 9:143-164

[141] Michael C C (2003), “Finding the Vocabulary of Program Behavior Data for Anomaly Detection”. In: Proc of the DARPA Information Survivability Conference and Exposition, 2003. pp 152- 163

[142] Michael C C, Ghosh A (2002), “Simple, State-Based Approaches to Program-Based Anomaly Detection”. ACM Transactions on Information and System Security, 5(3):203-237

[143] Mitchell T M (1997), *Machine Learning*. WCB/McGraw-Hills.

[144] Mukherjee B, Heberlein L T, Levitt K N (1994), “Network Intrusion Detection”. IEEE Network, 8(3):26-41

[145] Mukkamala R, Gagnon J, Jajodia S (2000), “Integrating Data Mining Techniques with Intrusion detection Methods”. In: Research Advances in database and Information System Security: IFIPTCII, 13th working conference on Database security, July, USA, Kluwer Academic Publishers.

[146] Mukkamala S, Janoski G and Sung A (2001), “Intrusion Deetection Using Neural Networks and Support Vector Machines”. In: Proc of Proc of International Joint Conference on Neural Networks (IJCNN’01), vol 2. pp 1853-1858

[147] Nash D A, Ragsdale D J(2001), “Simulation of Self-similarity in Network Utilization Patterns”. Transactions of the IEEE Systems, Man, and Cybernetics, Part A, 31(4):327-331

[148] Nicholas C, Dahlberg R (1998), “Spotting Topics with the Singular Value Decomposition”. In: Proc. PODDP’98, LNCS# 1481, Springer-Verlag. pp 82-91

[149] Noel S, Wijesekera D, Youman C (2000), “Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt”. In Barbar D, Jajodia S (eds.) *Applications of Data Mining in Computer Security*. Kluwer Academic Publisher. pp 2-30

[150] Okamoto T, Ishida Y (1999), “Multiagent Approach Against Computer Virus: An Immunity-Based System”. In: Proc of the AROB’99, Oita, Japan. pp 69-72

[151] Pawlak Z(1991)“ Rough sets: Theoretical aspects of reasoning about data”. Kluwer Academic, Dordrecht.

[152] Paxson V (1999), “Bro: A System for Detecting Network Intruders in Real-Time”. Computer Networks, 31(23-24):2435-2463

[153] Pong Chan K, Fu A W C (1999), “Efficient Time Series Matching by Wavelets,” In: the Proceedings of ICDE. pp 126-133

[154] Porras P A (1993), “STAT – A State Transition Analysis Tool For Intrusion Detection”. Technical Report, Number TRCS93-25, Computer Science. University of California, Santa Barbara.

[155] Pouzol J, Ducass M (2002), “Formal Specification of Intrusion Signatures and Detection Rules”. In: Proc of the 15th IEEE Computer Society Foundations Workshop (CSWF). Cape Breton, Nova Scotia, Canada. pp 64-74

[156] Qin M, Hwang K (2004), “Anomaly Intrusion Detection by Internet Datamining of Traffic Episodes”. Submitted to: ACM Transactions on Information and System Security (TISSec) 2004.

[157] Ramadas M, Ostermann S, Tjaden B C (2003), “Detecting Anomalous Network Traffic with Self-organizing Maps”. In: Proc of the Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003, Pittsburgh, PA, USA. LNCS#2820 Springer 2003. pp 36-54

[158] RSES: Rough Set Exploration System. http://alfa.mimuw.edu.pl/ rses/

[159] Rigoutsos I, Floratos A (1998), “Combinatorial Pattern Discovery in Biological Sequences”. Bioinformatics, 14(1):55-67

[160] Sabhnani M, Serpen G (2004), “ On Failure of Machine Learning Algorithms for Detecting Misuse in KDD Intrusion Detection Data Set”. Accepted: Journal of Intelligent Data Analysis, 2004.

[161] Saia J (2000), "Spectral Analysis for Information Retreival and Data Mining". General paper, submitted to the University of Washington, Computer Science and Engineering Department.

[162] Sastry Ch S, Pujari A K, Deekshatulu B L Bhagvati C (2004), "A Wavelet Based Multiresolution Algorithm for Rotation Invariant Feature Extraction," To appear in: Pattern Recognition Letters.

[163] Sato I, Okazaki Y, Goto S (2002), "An Improved Intrusion Detection Method Based on Process Profiling". IPSJ Journal, 43(11): 3316-3326

[164] Schultz M G, Eskin E, Zadok E, Stolfo S J (2001), "Data Mining Methods for Detection of New Malicious Executables". In: Proc of IEEE Symposium on Security and Privacy. Oakland, CA.

[165] Schölkopf B, platt J, Shawe-Taylor, Smola A J, Williamson R C (2001), "Estimating the Support of a High-Dimensional Distribution". Neural Computation, 13(7). pp 1443-1471

[166] Sebring M M, Shellhouse E, Hanna M E, Whitehurst R A (1988), "Expert System in Intrusion Detection: A Case Study". In: Proceedings of the 11th National Computer Security Conference. pp 74-81

[167] Sekar R, Bowen T, Segal M (1999a), "On Preventing Intrusions by Process Behavior Monitoring". In: Proc of the USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, USA.

[168] Sekar R, Guang Y, Verma S, Shanbhag T (1999b), "A High-Performance Network Intrusion Detection System". In: Proc of the 6th ACM Conference on Computer and communications Security Kent Ridge Digital Labs, Singapore. pp 8-17

[169] Sekar R, Gupta A, Frullo J, Shanbhag T, Tiwari A, Yang H, Zhou S (2002), "Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions". In: Proc of the 9th ACM Conference on Computer and Communications Security. Wyndham City Center, Washington, DC, USA.

[170] Seleznyov A, Mazhelis O (2002), "Learning Temporal Patterns for Anomaly Intrusion Detection". In: Proc of the 2002 ACM symposium on Applied computing, Madrid, Spain. pp 209-213

[171] Seleznyov A, Puuronen S (1999), "Anomaly Intrusion Detection Systems: Handling Temporal Relations between Events". In: Proc of 2nd Interantional Workshop on Recent Advances in Intrusion Detection, Lafayette, Indiana, USA.

[172] Seleznyov A, Terziyan V, Puuronen S (2000), "Temporal-Probabilistic Network Approach for Anomaly Intrusion Detection". In: Proc of 12th Annual Computer Security Incident Handling Conference, Chicago, USA.

[173] Sequeira K, Zaki M J (2002), "ADMIT: Anomaly-base Data Mining for Intrusions". In: Proc of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada. pp 386-395

[174] Sheikholeslami G, Chatterjee S, Zhang A (1998), “WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases”. In: the Proceedings of the 24rd International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. pp 428-439

[175] Shieh Shiuh-Pyng, Gligor V D (1997), “On a Pattern-Oriented Model for Intrusion Detection”. IEEE Transaction on Knowledge and Data Engineering, 9(4):661-667

[176] Somayaji A, Forrest S (2000), “Automated Response Using System-Call Delays”. In: Proc of the 9th USENIX Security Symposium, Berkeley. pp 185-198

[177] Somayaji A, Hofmeyr S, Forrest S (1997), “Principles of a Computer Immune System”. In: Proc of the New Security Paradigms Workshop, Langdale, Cumbria. pp 75-82

[178] Spafford E, Zamboni D (2000), “Data collection mechanisms for intrusion detection systems”. CERIAS Technical Report 2000-08, CERIAS, Purdue University, 1315 Recitation Building, West Lafayette, IN.

[179] Staniford S, Hoagland J A, McAlerney J M (2002), “Practical Automated Detection of Stealthy Portscans”. Journal of Computer Security, 10(1-2):105-136

[180] Stefanowski J (1998), “On Rough Set Based Approaches to Induction of Decision Rules”. In: Polkowski L, Skowron A (eds.) *Rough Sets in Data Mining and Knowledge Discovery*, Physica Verlag, Heidelberg. pp 500-529

[181] Sung A H, Mukkamala S (2003), “Identifying Important Features for Intrusion Detection Using Support Vector Machine and Neural Network”. In: Proc of the 2003 Symposium on Applications and the Internet (SAINT’03). IEEE Computer Society.

[182] Tan K, Maxion R (2002a), “’Why 6?’ Defining the Operational Limits of stide”. In: Proc of the IEEE Symposium on Security & Privacy. pp 188-201

[183] Tan Kymie M C, Killourhy K S, Maxion R A (2002b), “Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits”. In: Fifth International Symposium on Recent Advances in Intrusion Detection (RAID-2002). Andreas Wespi, Giovanni Vigna and Luca Deri (Eds.), 16-18 October 2002, Zurich, Switzerland,. LNCS # 2516, Springer-Verlag, Berlin. pp 54-73

[184] Tandon G, Chan P (2003), “Learning Rules from System Calls Arguments and Sequences for Anomaly Detection”. In: ICDM Workshop on Data Mining for Computer Security (DMSEC), Melbourne, FL. pp 20-29

[185] Thaler S L (2002), “AI for Network Protection: LITMUS - Live Intrusion Tracking Via Multiple Unsupervised STANNOs”. PC-AI, 16(1)

[186] Tidwell T, Larson R, Fitch K, Hale J (2001), “Modeling Internet Attacks”. In: Proc of the IEEE SMC Information Assurance Workshop, West Point, NY. pp 54-59

[187] Turner J S (1986), “New Direction in Communications (or Which to the Information Age?)”. IEEE Communications Magazine, 24(7):8-15

[188] UNM System Calls data set: http://www.cs.unm.edu/ĩmmsec/systemcalls.htm

[189] Wang H, Bell D (2004), "Extended k-Nearest Neighbours Based on Evidence Theory, To appear in: The Computer Journal.

[190] Warrender C, Forrest S, Pearlmutter B (1999), "Detecting Intrusions Using System Calls: Alternative Data Models". IEEE Symposium on Security and Privacy. pp 133-145

[191] Wespi A, Dacier M, Debar H (1999), "An Intrusion-Detection System Based on the Teiresias Pattern Discovery Algorithm". In: Proc of EICAR'99. Denmark.

[192] Wespi A, Dacier M, Debar H (2000), "Intrusion Detection Using Variable-Length Audit Trail Pattern". In : Proc of RAID 2000, Toulouse, France. LNCS# 1907. pp 110-129

[193] Willinger W, Taqqu M, Erramilli A (1996), "A Bibliographical Guide to Self-Similar Traffic and Performance Modeling for Modern High-Speed Networks". F. P. Kelly, S. Zachary and I. Ziedins (eds.), *Stochastic Networks*, Oxford University Press, Oxford, pp 339-366

[194] Xia X, Lazarou G Y, Butler T (2004), "Automatic Scaling Range Selection for Long-range Dependent Network Traffic". Submitted to: IEEE Communications Letters.

[195] Zhu D, Premkumar G, Zhang X, Chu Chao-Hsien (2001), "Data mining for Network Intrusion Detection: A comparison of alternative methods". J. Decision Sciences, 32(4):635-660