Y-MEANS CLUSTERING Vs N-CP CLUSTERING WITH CANOPIES FOR INTRUSION DETECTION

BY

SIVANADIYAN SABARI KANNAN

Bachelor of Engineering

Madras University

Chennai, India

2002

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfilment of
the requirements for
the Degree of
MASTER OF SCIENCE
December 2005

Y-MEANS CLUSTERING Vs N-CP CLUSTERING

WITH CANOPIES FOR INTRUSION

DETECTION

Thesis Approved:

Dr. JOHNSON P. THOMAS
Thesis Advisor

Dr. G.E.HEDRICK

Dr. DEBAO CHEN

Dr. GORDON EMSLIE
Dean of the Graduate College

ACKNOWLEDGEMENTS

There are many people who I would like to thank through out my graduate experience, but two stand out as the most noteworthy and significant. These are my advisor Dr. Johnson P. Thomas and Dr. Blayne E. Mayfield, who taught me the techniques of programming.

The mediocre teacher tells.
The good teacher explains.
The superior teacher demonstrates.
The **great** teacher inspires.
-William Arthur Ward.

Both the teachers inspired me. I thank my advisor for his continuous support throughout my thesis work. I never had an easy question, but my advisor always had an easy answer. He was always there to listen and give advice. He showed me distinctive ideas to approach a research problem and the need to be determined to realize my goals. Without Dr. Mayfield, I would not have come this far as far as software programming is concerned. I also thank my other committee members, Dr. G. E. Hedrick and Dr. Debao Chen for their timely suggestions and comments. I give my special thanks to Dr. Hedrick who asked me good questions and enlightened me, when I had difficulties in answering them during my final thesis presentation.

I have no words to express my gratitude toward my family: my parents Kannan and Visalakshi for giving me life in the first place, for educating me with aspects from both arts and sciences, for providing unconditional support and motivation to pursue my interests. My brother, Shankkar for listening to my complaints and frustrations through out my Masters program. Last, but not least, I thank my friends, who were always with me during my tough times and worst struggles.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Introduction

Computer systems and networks have been shown to struggle from security vulnerabilities and openness, regardless of their manufacturer, or origin. It is both technically difficult and expensive to ensure that an information system will not be harmed by attacks exploiting those vulnerabilities. This is the reason that intrusion detection systems (IDSs) are vital to supervise and scrutinize systems for intruders during their lifetime and to detect possible attacks against them [1]. A very wide range of activity falls under this definition, including attempts to de-stabilize the network as a whole, gain unauthorized access to files or privileges, or simply mishandling and misuse of software.

IDSs are the only means of detecting and responding to hostile attacks in a reasonable amount of time. IDSs allow for complete monitoring of modern networks, giving an organization real-time insight into threats to information systems [1]. Without an IDS, an organization could be repeatedly attacked and compromised without anyone realizing. IDSs are a non-invasive technology. If properly configured, they cannot harm or disrupt normal business activities. Other security technologies (like firewalls) can be

single points of failure that add significant risk when implemented. Below we examine the different genres of IDS [1, 2].

Threats to network systems come typically from the malfunction of hardware or software, or through malicious behavior by users of software. Promptly resolving network incidents is very important, considering the huge costs of data loss and system downtime [2, 3]. The abundance of computational resources makes the lives of computer hackers easier. Without much effort, they can acquire detailed descriptions of system vulnerabilities and exploits to initiate attacks accordingly. Statistics from CERT Coordination Center (CERT/CC) [4], the most influential reporting center for internet security problems, show that there was a dramatic increase of reported network incidents to CERT/CC from 1988 to 2002, as illustrated in figure 1. This trend is expected to continue, as the number of incidents in the first two quarters of 2003 reached 76,404, nearly the total number of 2002.

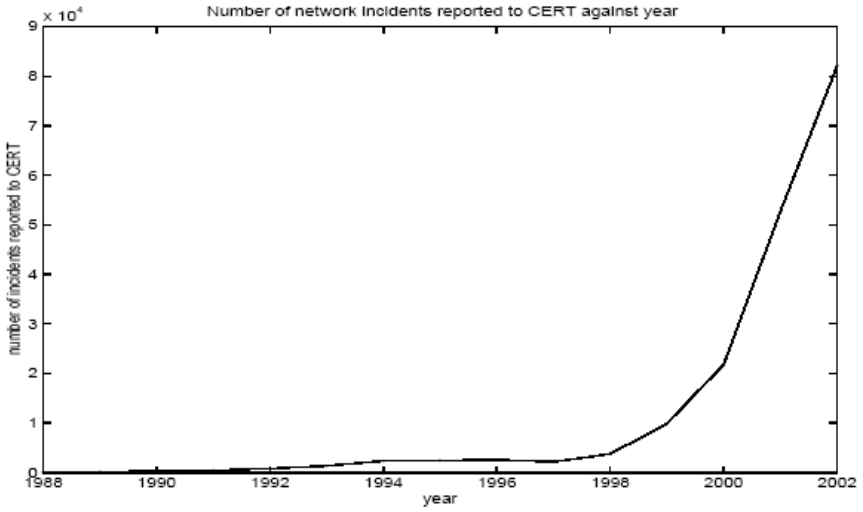

Figure 1: The number of incidents reported to CERT/CC from year 1988 to 2002. (The statistics are from http://www.cert.org/stats/cert stats.html#incidents. Please note that this reference is from the web, which may not be authoritative enough)

One should be aware of the fact that IDSs are not "THE" solution to the entire network and organization threats and no security technology can be considered in that way. This is because new attacks and intrusions evolve every other day and the IDS technology should be made to learn and recognize these threats. The catalog of all known attacks rapidly changes making this process of signature learning an intimidating task for the IDS technology developers.

In this work, we present a clustering framework for anomaly detection, where we plot all the data points in a complete metric space. We then manipulate these data points to create a learning pattern of structures called canopies. With this arrangement, we determine the outliers by taking into the consideration their position and the region they occupy. This requires a complex and costly pair-wise distance computation, which becomes a tedious and time-consuming task when the dataset produces a high dimensional metric space and an enormously large number of elements. We address this complexity with a solution called "Canopies". Canopies are the substructure of clustering which lie in the metric space with the possibility of overlapping patterns. We have devised an algorithm to work on these canopies to cut down the search and trim down the costlier pair-wise distance computation substantially. The major and notable advantages of our algorithm are its simplicity and running time efficiency. The devised algorithm also eliminates the shortcomings of K-means algorithm which will be explained in a later chapter. It also has the advantage of learning new intrusions by the use of proper training process. We have compared our algorithm with Y-means clustering to depict the detection and false alarm rate efficiency.

In subsequent chapters, we present a detailed overview of IDS followed by the categorical review of various clustering methods used in the past and a brief explanation of their working. We next explain the K-means algorithm and its shortcomings followed by a description of the Y-means clustering algorithm. In chapter 5, the basic background idea, methodology, dataset description and detailed explanation of our algorithm is presented. This is complemented by the findings and the results of our work depicted by a Receiver Operating Characteristics (ROC) curve.

CHAPTER II

INTRUSION DETECTION SYSTEMS (IDSs) – A PREVIEW

2.1 Types of IDSs

IDSs have matured to the point where there are essentially two types of IDs: Network IDS (NIDS) and Host IDS (HIDS). Host IDS resides on one machine and monitors that specific machine for intrusion attempts. More popular is the Network IDS, which monitors traffic as it flows through a network en route to other hosts. One type is not better than the other; each is appropriate for specific situations.

2.1.1 Host-Based IDS

Host-based IDSs (HIDSs) monitor for attacks at the operating system, application, or kernel level. HIDSs have access to audit logs, error messages, service and application rights, and any resource available to the monitored host. Additionally, HIDSs can be application aware. They have knowledge about what normal application data looks like, and what abnormal data looks like. They can monitor application data as it is being decoded and manipulated by the actual application. The benefits that HIDSs enjoy stem from this privileged access to the host [5].

HIDSs are better able to determine whether an attack was successful. Malicious traffic looks remarkably similar to normal traffic, for this reason NIDSs are notorious for creating false alerts. On the other hand, HIDSs are more accurate at detecting genuine intrusions because they do not generate the same volume of false positives as a NIDS. HIDSs leverage their privileged access to monitor specific components of a host that are not readily accessible to other systems. Specific components of operating systems, such as passwd files in UNIX and the Registry in Windows, can be watched for misuse [5, 6]. There is too great a risk in making these types of components available to a NIDS to monitor.

HIDSs are in tune with the host they reside upon. They have deep knowledge that is available only to an IDS that actually resides on the same computer that is being monitored. Therefore, HIDSs can have specific knowledge about the host and the type of activity that is normal for it. Traffic sent to the host might appear perfectly normal to a NIDS, but be recognized by the HIDS as abnormal and malicious. For this reason, HIDSs can discover attacks that a NIDS would not be able to.

Host-based IDSs do have some significant disadvantages. Because they reside on the monitored host, they have a limited view of the entire network topology. HIDSs cannot detect an attack that is targeted for a host that doesn't have an HIDS installed. An attacker can compromise a machine that lacks an HIDS and then use legitimate access to a protected machine, and the HIDS would be none the wiser. To monitor for intrusion attempts, the HIDS has to be placed on every critical host. This becomes cost prohibitive as the number of hosts critical to the organization grows. Running IDSs at the host level also means that you need to have an HIDS version available for every operating system

we need to protect. If we have obscure versions of operating systems at any organization or run legacy systems, we may not be able to provide the coverage even if your organization can afford it [5].

HIDSs that rely on audit logs and error messages are essentially detecting attacks after they have occurred, which can lead to all sorts of problems. Some attacks can compromise the host before data is written to a log, effectively disabling the HIDS. HIDSs rely on the host to facilitate communication to the intrusion analyst; therefore any attack that can disable the host outright goes unnoticed [5, 6].

2.1.2 Network-Based IDS

Network IDSs (NIDSs) are placed in key areas of network infrastructure and monitor traffic as it flows to other hosts. Network based IDS has grown in popularity and outpaced the acceptance of HIDS. A Network IDS is more cost effective than an HIDS because it can protect a large swath of network infrastructure with one device. With NIDS, the intrusion analyst has a wide-angle view of what is happening in and around the network. Monitoring for specific hosts or attackers can be increased or decreased with relative ease [6].

A NIDS can be more secure and less prone to outages than an HIDS. The NIDS should be run on a single hardened host that supports only services related to intrusion detection, making it more difficult to disable. NIDSs lose the disadvantages of relying on the integrity and availability of the monitored host, and are subsequently less prone to unobserved outages.

By not relying on the security of the host, NIDSs are not as prone to evidence destruction as HIDSs. Because NIDSs capture data and store it on a different machine, an attacker cannot easily remove the evidence of an attack.

NIDSs do have some disadvantages inherent in their design. NIDSs must be extraordinarily proficient at sucking up large amounts of network traffic to remain effective. As network traffic increases exponentially over time, the NIDS must be able to grab all this traffic and interpret it in a timely manner. Currently, NIDSs must be carefully placed and tuned to avoid situations where packet loss can occur. This can often require placing several NIDSs downstream from a core router or switch [6, 7].

NIDSs are also vulnerable to IDS evasion techniques. Hackers have discovered numerous methods for hiding malicious traffic in ways an NIDS cannot detect. One such method takes advantage of the process that occurs when a network connection exceeds the maximum allowable size for a packet. This is called *fragmentation*. When the host receives these fragmented packets, it must reassemble them to correctly interpret the data. Different operating systems reassemble the packets in different orders. Some start with the first packet and work forward, whereas others do the reverse. Reassembly order is insignificant if the fragments are consistent and do not overlap as expected. If the reassembly overlaps, the results will differ from each other, depending on the reassembly order. Choosing the correct reassembly order to detect a fragmentation attack can be problematic for NIDSs [7].

Another method of IDS evasion is far simpler. Because a NIDS captures traffic as it traverses a network, security measures intended to thwart eavesdropping can prevent a NIDS from doing its job. Encrypted traffic is often use to secure Web communication

and is increasingly becoming the norm for delivering confidential information. Attackers can use this to their advantage by sending attacks in encrypted sessions, effectively hiding their exploit from the NIDS's watchful eye. Some NIDSs support features that decrypt traffic before the IDS engine interprets it, but this option opens up a new vulnerability that some organization may not be willing to accept [7].

2.1.3 A Mixed Approach

Both intrusion detection models ca be an effective component of a defense in depth when properly configured and maintained. An important point to remember is that we don't have to choose one flavor of IDS exclusively. A NIDS has advantages that enable it to protect large portions of network infrastructure reasonably well. An HIDS offers fine tuned protection for mission-critical hosts.

Most organizations start their foray into intrusion detection with an NIDS. After growing accustomed to intrusion detection they gradually place HIDSs on hosts that are critical to day-to-day operation. This methodology gives complete intrusion detection coverage for an organization [7].

2.2 Methods of Detecting Intrusions

IDSs have several methods of detecting intrusions at their proposal. Certain techniques are better suited to monitoring for different types of intrusions; IDSs are likely

to employ more than one variety of detection. Signature detection is the most accurate technique of detecting known attacks.

2.2.1 Signature Detection

Signature detection identifies security events that attempt to use a system in a non-standard means. Known representations of intrusions are stored in the IDS and are then compared to system activity. When a known intrusion matches an aspect of system use, an alert is raised to the IDS analyst.

Known representations of intrusions are termed *signatures*. Signatures must be created to exactly match the characteristics of a specific intrusion and no other activity to avert false positives. In an NIDS, a specific signature is created that matches either the protocol elements or content of network traffic. When the NIDS detects traffic that matches the signature, an alert is crafted. The Large ICMP Packet Remote Denial of Service (DoS) attack for Internet Security System's BlackIce Defender is an easy-to-understand example [7, 8].

BlackIce Defender is a common personal firewall for home and small business use. A security researcher found that sending an unusually large ICMP packet to a machine protected by BlackIce would cause that machine's remote host to crash. To detect attacks against BlackIce, a signature was created to trigger on any ICMP packet over 10,000 bytes [8].

When a signature matches an intrusion, an alert is always generated. In addition, almost every type of malicious traffic can be identified by a unique signature. Therefore,

most malicious traffic can be caught by an IDS using signature detection, but they are a small minority and can be detected by other means.

Signature detection does have some limitations. It has no knowledge of the intention of activity that matches a signature; hence it triggers alerts even if the traffic is normal. Normal traffic often closely resembles suspicious traffic; hence NIDSs that use signature detection are likely to generate false positives [7, 8].

Signature detection requires previous knowledge of an attack to generate an accurate signature. This fact makes an IDS that utilizes signature detection as its only means of monitoring blind to unknown attacks or attacks without a precise signature. In some cases, the modification of a single bit is enough to cause an IDS to miss an attack [7, 8].

New attacks require new signatures, and the rising tide of vulnerabilities ensures that the signature bases will grow over time. Every packet must be compared to each signature for the IDS to detect intrusions. This can become computationally expensive as the amount of bandwidth increases. When the amount of bandwidth overwhelms the capabilities of the IDS, it causes the IDS to miss or drop packets.

2.2.2 Anomaly Detection

Anomaly Detection detects misuse by measuring a norm over time and then generating an alert when patterns differ from the norm. Anomaly detection comes in many different forms.

Anomaly detection can be used at the application level to monitor the activity of users. The anomaly detection IDS gathers a set of data from the system activity of the user. This baseline dataset is then deemed “normal use”. If the user deviates from the normal use pattern, an alarm is raised. It can be used to monitor for privilege escalation attacks. If a normal user account does not have privileged access to an important operating system file, such as the SAM file in Windows operating systems, but is seen to be accessing it readily, the IDS determines that potentially damaging activity has taken place and generates an alert [9].

An anomaly detection IDS is more adroit at catching, sophisticated attackers. An attacker can replicate a signature matching IDS in a controlled environment. The attacker can test out potential intrusions and discover which ones the signature matching IDS will notice. With an anomaly detection IDS, however, the attacker cannot predetermine which intrusive activity will go unnoticed.

The key benefit of anomaly detection IDSs is that they do not rely on having previous knowledge of an attack. As long as the IDS can determine that the attack differs significantly from normal use, it can detect the attack.

Like signature detection, anomaly detection has some limitations as well. The training period presents a problem for this method of monitoring for malicious use. You must assume that the data collected in the baseline dataset is not malicious and is normal activity. Anomaly detection can be prone to a relatively high degree of false positives. Suppose a particular type of traffic is rare, but non-malicious and normal. If this traffic was not captured when the IDS was generating baseline data, a false positive would be generated when the IDS encountered the traffic. This is a major problem, because over

time network traffic is composed of significant amounts of randomly occurring rare data. This makes anomaly detection not as accurate and hence not as popular as signature detection.

2.2.3 Integrity Verification

Integrity verification is a simple but highly effective means of monitoring for intruders. It works by means of generating a checksum for every file on a system, and then periodically comparing that checksum to the original file to ensure a change has not occurred. If an unauthorized file change transpires, an alert is generated.

A large number of files on any system regularly change in the course of normal operation. The integrity verification IDS must be carefully turned to avoid false positives. The checksums need to be reset when legitimate changes occur.

Integrity verification can be used to detect Web page defacements. Attackers often gain access to unpatched external facing Web servers and change the content the Web server displays. An integrity verification IDS could be deployed to create checksums and monitor specific Web page files. When the attacker changes the Web page's content, the checksum verification fails and the appropriate party is notified. The files on an external facing Web site should not change frequently enough to create a deluge of false positives. In addition, the IDS can be configured to automatically rollback the file to its unaltered state.

Integrity verification has some limitations as well. The primary disadvantage with integrity verification technology is that it requires access to sensitive files on the

monitored host. This dictates that it be a strictly host-based IDS, meaning that it inherits all the inefficiencies and drawbacks of an HIDS. In addition, the checksums can be altered to match the adulterated original file, rendering the integrity verification IDS useless. Storing checksums on a dedicated, hardened server can reduce the risk of this occurring, but does not completely eliminate it [9].

2.3 Origin of Attacks

Threats to information resources come in a variety of forms. Security of information can be compromised by very simple means. Although there are many threats to digital infrastructure, this section focuses on network-borne threats that an IDS is designed to monitor for.

Network-based threats can be separated into two categories:

- Internal Threats
- External Threats

Attack origins are important to the field of intrusion detection. We must know where attacks are initiating from to deploy intrusion monitoring in the most effective locations. A common statistic is that almost 80% of successful attacks are internal [9].

2.3.1 External Threats

One way of looking at the 80% statistic is that organizations are doing a pretty good job of protecting from external threats. It is likely that the vast majority of

attempted attacks are orchestrated from the external side and not the internal. The overwhelming majority of these external attacks are unsuccessful, whereas most internal attacks are executed with some degree of success.

This is not to downplay the risks external to an organization. It takes only one small chink in the armor of an external defense to allow significant damage. A single remotely exploitable host, be it a router, firewall, mail server, or any other externally facing device, can cause serious harm. Although the compromised host may not be of great value itself, an attacker can leverage access to the host to penetrate deeper within the security layers. Attackers frequently utilize compromised externally facing hosts to access internal devices that have less stringent security controls.

External security is often overlooked at organizations that feel they are not visible public targets. Small- or medium- sized organizations make the mistake of thinking they are not important enough for a hacker to target them. A hacker frequently scans the Internet looking for hosts vulnerable to exploit code the hacker has previously acquired or developed. The hacker is chiefly concerned with making use of a new exploit, instead of actively targeting a host [9].

2.3.2 Internal Threats

Internal attacks represent the majority of successful attacks on network infrastructure. Internal attacks can be damaging and far more difficult to discover. One factor that aggravates the situation is company insiders having extensive working knowledge of security controls and ample time to plan an attack. Insiders can leverage

the legitimate access they already possess to gain unauthorized additional access to systems.

Internal attacks are more difficult to detect than external attacks. This happens when organizations are not monitoring the inside as heavily as the outside. An internal attack may be the result of an employee gradually accumulating privileged access and information over a period of years or decades.

The internal infrastructure can also be unintentionally opened up to threats by uneducated or unsuspecting employees. Users can compromise internal security through the installation of firewall- defeating peer to peer (P2P) file sharing and instant messenger applications. Some P2P applications are packaged with spy ware or features that silently enable the sharing of the entire hard drive. Proxy-aware instant messengers, such as AOL Instant Messenger, can be used to slice through any open port on a corporate firewall. Modern viruses are bundled with numerous attack payloads that can open a system for the taking. Most non-technical users may be unaware that they are creating a gaping security hole by going about their daily activity.

An IDS on the internal side can be used to detect both intentional internal attacks and corporate policy violations. They can detect the signature of most P2P tools, inappropriate Internet usage, and instant messengers. This is in addition to the expected intrusion monitoring capability. These abilities make an internal IDS an extremely powerful security application [9].

The line between internal and external is increasingly blurred by corporate partnerships and the extranets that enable them. An attacker can hop from one extranet to another, making the source of an attack difficult to discern. As more and more internal

security breaches are discovered, organizations will seek to increase internal security in the future.

CHAPTER III

LITERATURE REVIEW

3.1 Cell-Based Clustering

Clustering is a well studied problem. However, the majority of work done is intended to optimize clustering. While some do provide a mechanism to handle noise but only to the extent of milding its affect on the overall quality of clusters. In more recent work, clustering (or a closely related approach) has been used to locate outliers in datasets [10-13]. An outlier is defined as an object with at least p fraction of the dataset is farther than distance D from the object, where p and D are parameters specified by the users [13]. They also propose an efficient cell-based approach for mining such outliers in high dimensional data. [12], [13] opt for a local perspective and investigate the problem of efficiently finding the top n outliers. They propose optimizations based on pruning those partitions that cannot contain any outliers. The notion of a "cell" is closely related to that of a cluster, and by their [12, 13] definition, dense regions cannot contain outliers. Interestingly, in our domain of network traffic some DOS attacks do generate large amounts of traffic and can possibly reside in dense regions.

Instead of a global perspective [13], Local Outlier Factor (LOF) [9] uses a local perspective and locates outliers with respect to the density in the local/neighboring

region. They illustrate the inability of conventional approaches to detect such outliers. LOF has two short-comings: one, their approach is very sensitive to the choice of *MinPts*, which specifies the minimum number of objects allowed in the local neighborhood (similar to *k* in K-means Neighbor Natives); second, and more importantly, their approach is not well-suited for very high dimensional data such as network traffic data.

[11] address the problem of mining outliers in high dimensional data. They calculate the sparsity coefficient, which compares the observed and expected number of data points, in "cubes" (spatial grid cells) generated by projections on the dataset. Their approach is computationally expensive, while they do provide theoretical examples of data points that are outliers in certain subsets of the feature space whose effect can possibly be offset by "noise" in the complimentary subset. They provide no practical example of such a scenario materializing; put another way they don't empirically illustrate discovery of any outlier which would have been missed by the current distance-based outlier paradigm.

[14] And [15] use clustering as an approximation of K-means Neighbor Natives to find sparse clusters and label them as anomalous.

3.2 Fuzzy Clustering for Intrusion Detection

Because anomaly detectors looks for abnormalities, many of the data mining techniques that seek to identify outliers in data become readily applicable. Hence many researchers have explored applying data mining techniques to the problem of intrusion detection. Lee et al., [16] performed experiments on *sendmail* system call data and network *tcpdump* data. They used RIPPER [17] [18] to generate classifiers for these

datasets. In another paper Lee et al., [19] describe how to use association rules and frequent episode algorithms to guide the process of audit data gathering and selection of useful features to build the classifiers.

Dokas et al., [20] have developed classification algorithms for intrusion detection. These algorithms are designed especially for learning from datasets in which the class of interest (i.e. the intrusion class) is significantly smaller than the class representing normal behavior. In this body of work, the authors discuss various outlier detection schemes for detecting network intrusions.

Dickerson et al., [21] developed the Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy sets and fuzzy rules. FIRE uses the *Fuzzy C-Means Algorithm* developed by Bezdek [22] to generate fuzzy sets for every observed feature. The fuzzy sets are then used to define fuzzy rules to detect individual attacks. FIRE does not establish any sort of model representing the quiescent state of the system, but instead relies on attack specific rules for detection.

3.3 Special Purpose IDSs and Anomaly Detection

The intuitively most appealing way of dealing with false positives is to build "better" IDSs, which trigger less false positives. This is a challenging endeavor because false positives are the result of multiple problems, including a lack of suitable audit sources [23,24], harsh real-time requirements (which preclude a thorough analysis of the audit data) [24,25], the problem that for some events (e.g., failed logins) it is undecidable whether they constitute attacks [26,27], and the inherent difficulty of writing correct

intrusion detection signatures [28-31]. A "better" IDS would have to address all these issues, and a small number of research projects have attempted to do so. Examples of IDSs that are less prone to false positives include the embedded detectors technology by [32], a lightweight tool for detecting Web server attacks by [33], and a network-based IDS that focuses exclusively on low-level network attacks [34]. Interestingly, all three IDSs share two commonalities: first, they have public signatures that can be tuned to a given environment, and second they are special purpose. *Special purpose IDSs* are tailored toward detecting one class of attacks (e.g., Web server attacks), and they monitor audit sources that are particularly suitable for this task. A drawback of special-purpose IDSs is that they must be combined with other complementary IDSs to obtain comprehensive coverage.

Clustering, or unsupervised learning, has attracted some interest [35–38] in the context of intrusion detection. The interesting feature of clustering is the possibility to learn without knowledge of attack classes, thereby reducing training data requirement, and possibly making clustering based techniques more viable than classification-based techniques in a real world setting. There exist at least two approaches. When doing *unsupervised anomaly detection* a model based on clusters of data is trained using unlabelled data, normal as well as attacks. The assumption is that the relative amount of attacks in the training data is very small compared to normal data, a reasonable assumption that may or may not hold in the real world context for which it is applied. If this assumption holds, anomalies and attacks may be detected based on cluster sizes. Large clusters correspond to normal data, and small clusters possibly correspond to attacks. A number of unsupervised detection schemes have been evaluated on the KDD

data set with varying success [35-37]. The accuracy is however relatively low which reduces the direct applicability in a real network.

In the second approach, which is denoted simply as *(pure) anomaly detection* in this paper, training data is assumed to consist only of normal data. Munson and Wimer [38] used a cluster based model (Watcher) to protect a real web server, proving anomaly detection based on clustering to be useful in real life.

3.4 Autoclass Bayesian Clustering

Autoclass [40, 41] is a Bayesian Clustering program developed by Peter Cheeseman and his colleagues at NASA [39]. It automated the process of model selection as well as the process of parameter estimation. By calculating the approximation of the marginal density of data after the integration of the parameters, Autoclass compares different models and uses Ocam Razer to favor models with less complexity.

3.5 Clustering and Classification Algorithm- Supervised (CCA-S) [42]

Clustering and Classification Algorithm – Supervised (CCA-S) is a data mining algorithm, which is developed for detecting intrusions into computer network systems for intrusion detection. CCA-S is used to learn signature patterns of both normal and intrusive activities in the training data and to classify the activities in the testing data as normal or intrusive based on the learned signature patterns of normal and intrusive

activities. CCA-S differs from many existing data mining techniques in its ability in scalable, incremental learning.

In CCA-S, a data record is considered as a data point in a p dimensional space. Each dimension is either a numerical or a nominal variable, called predictor variable, representing one attribute of the data. Each data point has also a label indicating the class of the data record, called the target variable. For computer intrusion detection based on signature recognition, the target variable is a binary variable with two possible values: 0 for normal and 1 for intrusive. CCA-S clusters data points based on two criteria: the distance between data points, and the class label of data points. Only data points that are close and same in their class label can be grouped together to form a cluster. Each cluster represents a signature pattern for normal activities or intrusive activities, depending on the class label of the data points in the cluster. Formally, each data point is a (p+1)-tuple with the attribute variable vector X containing the *p* dimensions of predictor variables and one target variable - Y. The training data set has N data points.

Step 1: Training (supervised clustering)

It takes mainly two steps to incrementally group the N data points in the training data set into clusters.

1. Scan the training data and compute the relative importance of each prediction variable with respect to the target variable. This step calculates the coefficient of the correlation between each predictor variable Xi and the target variable Y. In addition, two dummy clusters, one for normal activities and another for intrusive activities, are created. The centroid of the dummy cluster for normal activities is denoted by the mean vector of all the data points for normal activities in the training data set. The centroid of the

dummy cluster for intrusive activities is denoted by the mean vector of all the data points for intrusive activities in the training data set.

2. Incrementally group each point in the training data set into clusters. Given a data point X, we find the nearest cluster L to this data point using a distance metric weighted by the correlation coefficient of each dimension. If L has the same class label as that of X, we group X with L; otherwise, we create a new cluster with this data point as the centroid of the new cluster.

We then repeat the above steps until we process all the data points in the training data set.

Step 2: Classification

There are two methods to classify a data point X in a testing data set.

1. Assign the data point X the class dominant in the k nearest clusters which are found using a distance metric weighted by the correlation coefficient of each dimension; or

2. Use the weighted sum of the distances of k nearest clusters to this data point to calculate a continuous value for the target variable in the range of [0, 1].

Step 3: Incremental update

The statistics from the correlation and the clustering are stored. When new training data become available, each step of the training can be repeated for new data points to update the clusters incrementally. An estimate of the computation cost of CCA-S is provided here. Given N data points, and the total number of the resulting clusters L, the computation cost for training is O (p*N*L). And the computation cost of classifying a data point during testing is O (p*L). Hence, CCA-S is scalable to even large amounts of training data. This tremendously increment the computation cost of the cluster algorithm.

This can only be computed only when N data points and resulting clusters L are given.

3.6 Graph Clustering and Graph Drawing [43]

A graph is used as an abstraction of network traffic. Computers are represented by nodes, communication between computers is indicated by edges, weighted by the amount of exchanged data. Figure 2 shows such a clustering of a network.

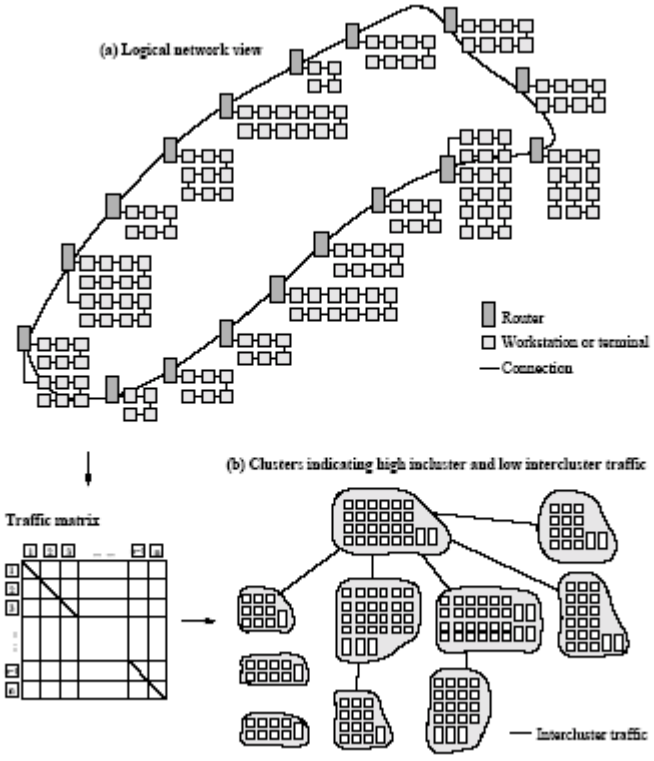

Figure 2: Communication clusters identified in the traffic matrix of a computer network.

In this system, graph clustering algorithms are applied to the traffic graph. Clustering decomposes a graph G= (V, E) into clusters Ci Є V, 0≤ i ≤ n-1, where C1 UC2

U... U Cn = V and for all $0 \leq i, j \leq n-1$: $Ci \cap Cj = \emptyset$. This decomposition should represent the internal structure of the graph. In the domain of network traffic, a cluster should consist of nodes with a high inter-node traffic.

There are several possibilities to present the results of the graph clustering. One approach is to present the results of the clustering process in lists. This form of presentation has several disadvantages. It is not possible to achieve a general overview on the current traffic structure in the network. Modifications of the structure are hard to discover in the lists. Due to this, a graphical representation of the clustering results was chosen.

Visualization of network traffic is an important task for planning and managing large networks. Research is done on this item to provide network managers with an insight to the usage of their systems. Most of these systems present traffic information in a geographical way. The visualization of the traffic depends on the location of the nodes. In this system, they've chosen to present the results of the traffic analysis in another way. Their focus is on the structure of the traffic, and therefore they group nodes with strong communication relations together without considering their geographic location.

This tool first places the clusters on the plane. The reader may note that the clusters form a new graph. Two clusters are connected if nodes within those clusters are connected by an edge. The clusters can be positioned using the Spring-Embedder. In a second step the nodes with the clusters are placed. This visualization helps the security manager to build his own opinion on messages from the event generator described in the next section. A major problem in intrusion detection systems are so called false negatives and false positives. The problem of any anomaly detection system is the fact, that

anomalies in the behavior are not necessarily based on intrusions (causing false positives) and intrusions do not necessarily cause anomalies in the system (leading to false negatives). This visualization helps discovering false positives (alarms without attack). It is easy to see, that there are more reasons for modifications in the traffic structure. Changes in network topology, new network devices, start or end of projects are examples for reasons for modifications in the typical structure.

The user of the visualization module is able to use his additional personal knowledge to decide on the results of the event generating process. A special benefit of this visualization is the possibility of displaying modifications of the traffic structure in consecutive traffic matrices. Position and color of nodes indicate changes in their membership of different clusters. It is easy to track varying cluster, giving information on nodes that have changed their communication behavior. It is even possible to present longer series of traffic matrix visualizations in the form of an animation, showing long term behavior of the communication structure.

CHAPTER IV

Y-MEANS CLUSTERING

4.1 Introduction

Y-means is one method clustering algorithm for intrusion detection. It is expected to automatically partition a data set into a reasonable number of clusters so as to classify the instances into `normal' clusters and `abnormal' clusters. It also overcomes the shortcomings of the K-means algorithm.

4.2 K-Means Algorithm [46]

The simple definition is that K-means clustering is an algorithm intended to categorize or congregate the objects based on features, attributes into K partitons or clusters, where K always represents a positive integer. The classification of data is the main purpose of the K-means clustering algorithm.

The process for K-means clustering and positioning the K centroids can be accomplished by the following procedure:

1. Position the K points in the feature space represented by the dataset that are being subjected to clustering. These plots set up the initial block of centroids

2. Allocate each subsequent data point from the considered dataset to the group that has the nearest centroid
3. In this manner, when all the points are assigned to the appropriate group, reassign the K centroids by computing the mean of that group.
4. Steps 2 and 3 are repeated until the movement of centroids ceases. This result in the partition of the datapoints into groups. The datapoints in a group represents some sort of similarity if they remain in that logical grouping.

K-means clustering suffers from two major shortcomings: number of clusters dependency and degeneracy. Number of cluster dependency is the value of k which has to be supplied by the user and is very critical to the clustering result. Degeneracy means that K-means clustering may end up in producing empty clusters that makes the clustering process to give poor computational scaling.

4.3 Y-Means Algorithm [37]

Figure 3 illustrates the Y-means algorithm. Similar to K means, it partitions the normalized data into *k* clusters. The number of clusters *k* can be a given integer between 1 and *n* exclusively, where *n* is the total number of instances. The next step is to find whether there are any empty clusters.

If there are, new clusters will be created to replace these empty clusters; and then instances will be re-assigned to existing centers. This iteration will continue until there is no empty cluster. Subsequently, the outliers of clusters will be removed to form new clusters, in which instances are more similar to each other; and overlapped adjacent

clusters will merge into a new cluster. In this way, the value of k will be determined automatically by splitting or merging clusters.

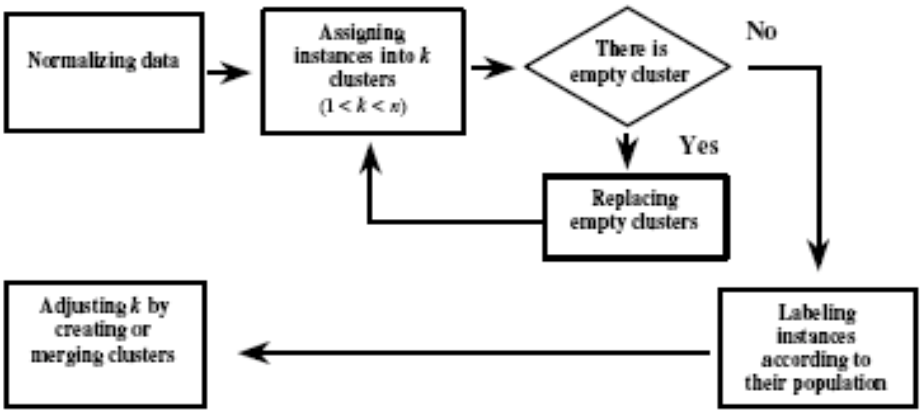

Figure 3: Y-Means Clustering Algorithm shown in a Flowchart

The last step is to label the clusters according to their populations; that is, if the population ratio of one cluster is above a given threshold, all the instances in the cluster will be classified as normal; otherwise, they are labeled intrusive.

4.4 Splitting Clusters

An outlier is a point that is quite different from the majority of the points in a cluster. When Euclidean distance is used to evaluate the difference between two points, an outlier is the point that is remote from the majority of points. Since the cluster center is the mean vector of all the points in the cluster, and all points are assumed to have equal weights, one can find outliers using the radius of points; that is, if the radius of a point is

over the threshold, it is considered as an outlier. The idea of detecting outliers comes from the theory of robust regression and outlier detection [7].

From the Cumulative Standardized Normal Distribution Function table in [2], it is found that 99% of the instances of the cluster stay within the circle with a radius of 2.32 σ, where σ is the standard deviation of the data. Therefore, the chosen threshold is t=2.32 σ. The area within the circle is called the *Confident Area* of the cluster. Thus, in the cluster any point that stays out of the *Confident Area* will be deemed an outlier, and the remotest outlier will be removed first to form a new cluster. Then, this new cluster may snatch some points from its neighbor clusters. In the Y-means algorithm, the splitting will iterate until no outlier exits. The splitting procedure turns clusters into finer grains; and makes the instances in the same cluster more similar to each other, while it increases the number of clusters. However, it may partition the data set into too small clusters, i.e., over-splitting. In order to avoid the over-splitting, we can merge the overlapped adjacent clusters.

4.5 Merging Clusters

When two adjacent clusters have an overlap, they can be merged into a larger cluster. The merging threshold is set to 2.32 σ as well; that is, whenever there are some points in a cluster's *Confident Area* also fall in another cluster's *Confident Area*, the two clusters can be merged. The center of the new cluster can be obtained simply by calculating the mean vector of the two previous centers.

CHAPTER V

METHODOLOGY

5.1 Data Set Assumption

The input to the algorithm is a large set of data with normal elements in majority and some attack data obscured within the data set. It can identify the unlabeled data and uncover the intrusions to the best possible. In general *Intrusion Detection Systems* (IDS) make two assumptions about the data set that trigger off the approach.

1. The number of normal data greatly surpasses the abnormal or attack data quantitatively
2. The attack data differs from normal data qualitatively

The algorithm will work only when the assumptions hold over the data. The intrusion detection algorithms will not be able to detect the malevolent behavior of an authorized user of the network who meticulously uses it in a legitimate procedure.

5.2 Data Representation

The data input for the algorithm is collected from the audit stream of the system. The audit data is represented as $x_1 \ldots x_m$. All possible input data are mapped to the space called inner product space H, which as a metric space, is complete. This space is a real

vector space of high dimension z. The use of using an inner product space is that we can define the dot product between elements in the inner product space at any instance. Hence we can also define distance as well as norm (what is norm?) on the space. This inner product space with an inner product < f, g > such that norm is defined by

$$| f |= \sqrt{< f, f >}$$

turns H into a **complete metric space.** If the metric defined by the norm is not complete, then H is instead known as inner product space. Let the input data element x_1 in the space H be represented as $H(x_1)$. To determine the distance between any two elements x and y in the input set, we can use the inner product space by determining the distance between H(x) and H(y) as

$$\text{Distance } (x, y) = | H(x) - H(y) |$$

$$= \sqrt{< H(x), H(x) > -2 < H(x), H(y) > + < H(y), H(y) >}$$

If the space is a Euclidean space then this distance represents the standard Euclidean distance in that space.

5.3 Data Normalization

This work deals with an enormously large data set with numerous attributes. It is difficult to determine weight values for the vectors representing attributes. It becomes inevitable to cut losses by performing normalization on the data. This will minimize the unnecessary mock weight on the attributes and will reduce the problem of biasing the concentration on one particular attribute with a naturally larger magnitude than the other attributes. Leaving the discrete and binary data that will not much influence the biasing,

continuous data should be taken care of by appropriate technique. The continuous data can be best normalized by replacing each attribute value with its distance to the mean of all the values for that attribute in the input region. In our case to do this normalization the mean and standard deviation vectors must be calculated:

$$mean_{\,n} = 1/\mathrm{m} \sum_{i=1}^{n} record_{\,n}$$

$$std.deviation_{\,\mathbf{n}} = \sqrt{1/(1-m)\sum_{i=1}^{i=m}(record_{\,n} - mean_{\,n})^{2}}$$

$$New\ record_n = (record_n - mean_{n)} / std.deviation_n$$

5.4 Basic Background Idea

This input data set for this work is inherently large in three senses at once - there are millions of elements, many thousands of features and many thousands of clusters. The requirement for efficient solution is a technique for clustering that is efficient when the problem is large in all of these three ways at once. Through a long literature survey and several clustering techniques, we found a close matching algorithm which resolves the problem of large data set to maximum extent. The technique is ***Efficient Clustering with Canopies,*** [45] where the key idea is to perform clustering in two stages, first a rough and quick stage that divides the data into overlapping subsets that are called "canopies", then a more rigorous final stage in which expensive distance measurements are only made among points that occur in a common canopy. This differs from other clustering methods in that it forms overlapping regions. We integrate this idea of forming canopies with K-

Means Algorithm with an optimizing technique which we call ***"Optimized N-Closest Points Algorithm"***.

This algorithm determines if a test data point lies in the sparse region of the inner product space H mentioned in the previous section by computing the sum of the distances to the n-closest points to the test data point and this calculation assigns the weight which will be referred as n-CP value. The inner product space H's dense regions will have numerous points close to each other and n-CP value will be less. If the point has high n-CP value then it means that attack element will be far from the normal elements in the inner product space which is checked by comparing with a threshold value. This idea will determine exactly how closely the test data point fits with already existing normal data in H space.

5.5 Optimizing Cluster

In this work, the canopy clustering technique is used specifically to chop down the search in the large set of data into smaller overlapping subsets or canopies which eliminates the necessity to check every data point. There should be a computational shortcut to eliminate the data in linear time from large set of cluster data. The clusters are used as a means to reduce the time consumed in finding the n-closest points. In other words they are hyper spheres that contain data from the training data set. Canopy clustering has a way to eliminate clusters without looking at the contents, thus drastically shrink the computation time.

In this work, the training data is clustered using fixed-width clustering algorithm. One variation would be placing the data point in exactly one cluster. If the cluster radius or cluster width is *r* and data are clustered in the vicinity of this radius, we can calculate the n-closest points from a given test point *p* using the mathematical property of inequalities. The data point *p1* and *p2* that are in the same cluster has the following properties:

$$\text{Dist}(p1, p2) \leq 2(\text{cluster radius})$$
$$\text{Dist}(p1, p2) \leq [\text{dist}(p1, \text{centroid}(p2)) + (\text{cluster radius})]$$
$$\text{Dist}(p1, p2) \geq [\text{dist}(p1, \text{centroid}(p2)) - (\text{cluster radius})]$$

5.6 Clustering Algorithm

The algorithm works in linear time to form the clusters from the input training data set T. The point p is a point for which the distance is calculated to fit n-Closest Points set nCP. C is the set of clusters. R is the cluster radius. D_i is the distance from p to the centroid of C_i.

5.6.1 Fixed Radius Clustering Algorithm

1. Set C = empty.
2. For all pt € T:
   (i) If there exists c_i such that $[D_i < R]$, add pt to the cluster c_i.

(ii) Else if C is empty or no c_i exists such that [D_i < R], build a new cluster c_{new} and insert pt as its centroid.

At the start, an empty cluster set C is formed. T is the training set and each point in the training set is taken into consideration in succession. A cluster radius R is decided for the algorithm and this will serve as the cluster width for the entire process. When a point pt from the training set T lies within the width of a cluster R, then that point is put in that cluster, which means that point is the member of that cluster. If there is an empty set of cluster or if the point falls out of the existing cluster within the cluster set C, then a new cluster is generated with width R and point pt is the centroid of that new cluster. Using this simple clustering algorithm, all the points in the training set are clustered.

5.7 Optimized n-Closest Points Algorithm

The canopy clustering overlapping strategy is followed here. The overlapped cluster boundary is carefully assessed using the Cluster Gap G.

In this algorithm:

n = Number of closest points required

pt = The point from which n closest points are need to be computed

C = Set of clusters

R = Cluster width

G = Cluster gap

nCP = Set of n closest points

C_p = Set of member points

Largest (nCP) = Element with the maximum value in the set nCP

Each cluster c from set C is comprised of the pair (c, D) where

c = Set of p points that lie within the cluster

D = Distance from the point pt to the centroid of cluster c

5.7.1 n-CP Clustering Algorithm

1. Let Cluster Gap G = 0, nCP value = 0, C_0 = Ø, C_p = Ø
2. For pt, compute D_i for each c_i € C
3. Rank the clusters in C by from lowest to highest.

   (C = {c_0, c_1, c_2, c_3, c_4...}
4. Do While (# (nCP) < n)

   (i) Take the closest cluster from the set C that is, first member c_0 of the sorted set C

   (ii) Add all the points in c_0 to C_0.

   (iii) Compute distances from test data point p to all the points in C_0 and drop them in C_p

   (iv) Test if [(D_1-R) <G] assign G = D_1-R

   (v) Else assign G= D_0+R

   (vi) Each p_i € C_p where distance (pt, p_i) < G

   a) If (# (nCP) < n) then insert from C_p to nCP

b) Else if (# (nCP) == n) and [distance (pt, p_i) < Largest (nCP)] then insert from C_p to nCP by replacing the Largest (nCP)

(vi) Eliminate c_0 from C

5. If nCP Value > IDS threshold broadcast it as an attack or intrusion else a normal data

First, to compute n-closest data points, the distance from a given data point pt to all the cluster centroid from the cluster set C is determined. From the data obtained from this computation, the clusters are ranked on the basis of their distances from lowest to the highest. The first cluster in this sorted set has the label of closest cluster c_0 in the set C and the next will be c_1, etc. Next computation is the distance from the point pt to all the points in the closest cluster c_0. This is shown in figure 4.These points belong to the set C_0 and they are added to the list of member points, C_p.

After these computations, the algorithm proceeds with a concern on the establishment of cluster gap G and the changes in the boundaries of the cluster. The whole point of this boundary modification is to encompass the least number of points to calculate the n closest points. The initial procedure of ranking the clusters results in a limited ordering. This is because the algorithm works on the concept of clustering with canopies and there could be overlapping. Drawing a boundary by means of cluster gap G ensures that no other point of the clusters other than those added to the nCP set is closer than the added ones. This optimizes the entire testing by checking only the points that ***should*** be tested are tested.

It is really significant to leave out the overlapped parts by other clusters because of the simple reason that they might include the points nearer to the point pt than the points in the closest cluster.

When the cluster gap G is first set up, the cluster c_0 is examined to see if it has overlapped the closer cluster c_1. If that gives a negative result then distance to the furthest point in the nearest cluster which will be (D_0+R) is lesser than that of the closest point in the next ranked cluster $c_{1\ n}$ and that will be (D_1-R). If the overlapping test is positive then that shows there exist overlapping clusters. In that case the cluster gap is set to D_i-R. This is the intersecting point nearest to point pt. All the points within this boundary are closer than any existing point in the cluster that has overlapped. This is shown in figure 5.

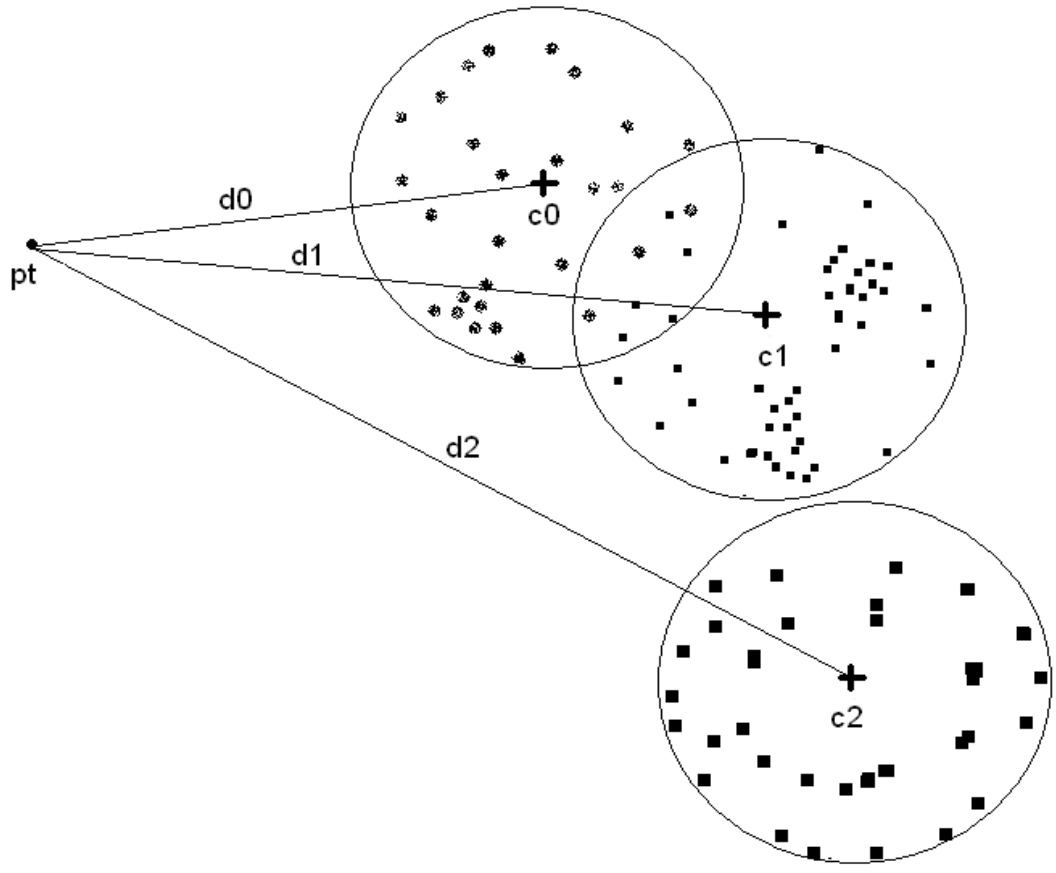

Figure 4: A cluster arrangement as a result of fixed width clustering algorithm

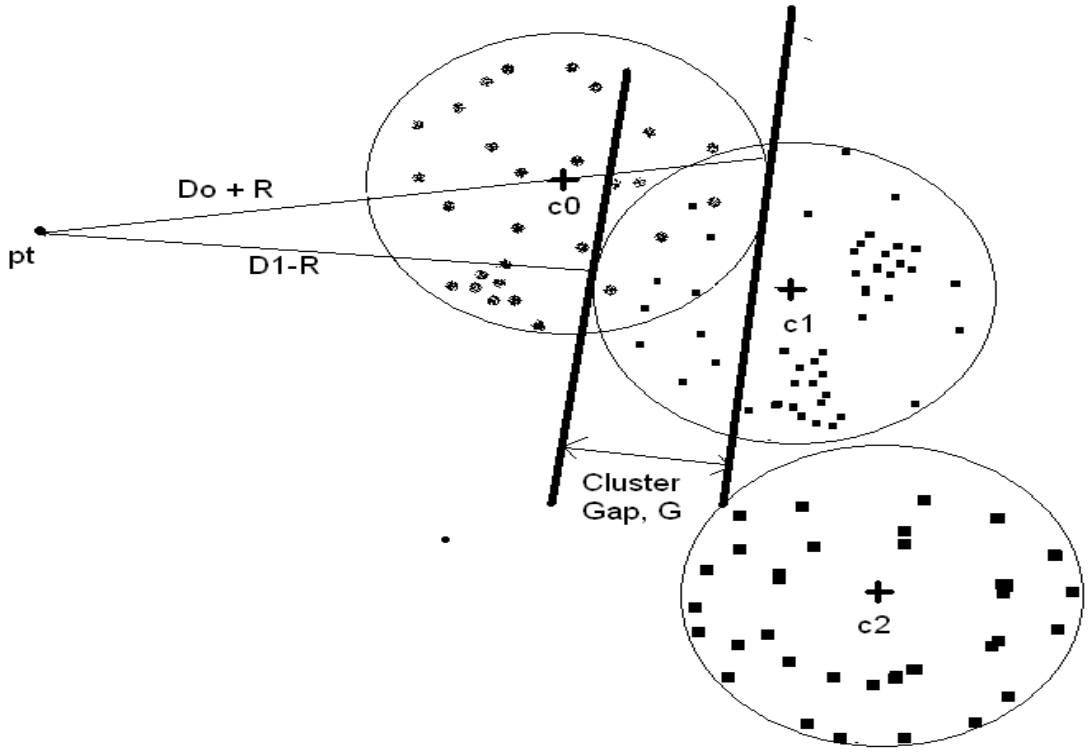

Figure 5: A cluster gap between the first two clusters in the ranked order

On setting up this boundary, all the points of C_p within the determined boundary are added to the nCP set. When the nCP set is maximized and the element with maximum value in nCP is greater than the given point than Largest (nCP) is removed and the given point is added. Then c_o is discarded by making c_1 the c_o and c_2 the new c_1. When the nCP set has n elements that will be the time to stop the iteration or else the iteration continues until the nCP is maximized with n points.

5.8 Dataset Descriptions

The proposed algorithm will use the network connection records from the KDD Cup 99 Data. The KDD cup data set can be obtained from *http://kdd.ics.uci.edu*

/databases/kddcup99/kddcup99.html. This includes ample number of intrusions of different kind simulated in a military network environment. It approximately consists of 4,900,000 data records. Each record is a vector of extracted feature weights collected from the raw network data during the simulated intrusions. A sequence of TCP packets to and from some IP addresses, starting and ending at some well defined times is said to be a connection. All the connections are labelled as either normal or exactly one kind of attack and these labels are assumed to be correct. The KDD data set was obtained by simulating a large number of different types of attacks, with normal activity in the background

The simulated attacks fall in one of the four categories namely:

- DoS – Denial of Service
- R2L – Unauthorized access from a remote machine
- U2R - Unauthorized access to super user or root functions
- Probing – Surveillance and other probing for vulnerabilities.

The anomaly algorithms used for intrusion detection are sensitive to the ratio of intrusions in the data set. If the number of intrusions is too high, each intrusion will not show up as anomalous. In order to make the data set more realistic many of the attacks are filtered so that the resulting data set consisted of 1 to 1.5% attacks and 98.5 to 99% normal instances. The KDD-99 data set has 41 fields to represent the network connection records and those fields are listed in the table 1.

Feature Name	Feature Type
duration	continuous
protocol_type	symbolic
service	symbolic
flag	symbolic
src_bytes	continuous
dst_bytes	continuous
land	symbolic
wrong_fragment	continuous
urgent	continuous
hot	continuous
num_failed_logins	continuous
logged_in	symbolic
num_compromised	continuous
root_shell	continuous
su_attempted	continuous
num_root	continuous
num_file_creations	continuous
num_shells	continuous
num_access_files	continuous
num_outbound_cmds	continuous
is_host_login	symbolic
is_guest_login	symbolic
count	continuous
srv_count	continuous
serror_rate	continuous
srv_serror_rate	continuous
rerror_rate	continuous
srv_rerror_rate	continuous
same_srv_rate	continuous
diff_srv_rate	continuous
srv_diff_host_rate	continuous
dst_host_count	continuous
dst_host_srv_count	continuous
dst_host_same_srv_rate	continuous
dst_host_diff_srv_rate	continuous
dst_host_same_src_port_rate	continuous
dst_host_srv_diff_host_rate	continuous
dst_host_serror_rate	continuous
dst_host_srv_serror_rate	continuous
dst_host_rerror_rate	continuous
dst_host_srv_rerror_rate	continuous

Table 1: Features in KDD Cup 1999 Network connection records.

CHAPTER VI

EXPERIMENTS

The experiments are performed over the KDD-99 data set. The network connection records (KDD-99 Data Set) are analyzed to determine the performance of the algorithm.

6.1 Performance measures

To evaluate the proposed algorithm two major indicators of performance: the detection rate and the false positive rate are taken in to consideration. The detection rate is defined as the number of intrusion instances detected by the system divided by the total number of intrusion instances present in the data set. The false positive rate is defined as the total number of normal instances that were incorrectly classified as intrusions defined by the total number of normal instances. These are good indicators of performance, since they measure what percentage of intrusions the system is able to detect and how many incorrect classifications it makes in the process. These values over the labeled data are calculated to measure performance.

The trade-off between the false positive rate and detection rates is inherently present in numerous machine learning methods. By comparing these quantities against each other we can evaluate the performance invariant of the bias in the distribution of

labels in the data. This is especially important in intrusion detection problems because the normal data outnumbers the intrusion data by a factor of 100:1. The classical accuracy measure is misleading because a system that always classifies all data as normal would have 99% accuracy.

A ***Receiver Operating Characteristic (ROC) Curve*** is an aggregate of the probability of false alarms and the probability of detection measurements The ROC curves are plotted depicting the relationship between false positive and detection rates for one fixed training/test set combination. ROC curves are a way of visualizing the trade-offs between detection and false positive rates.

6.2 Experimental Setup

For the selected data set, the data is split into two segments. One portion, the training set, was used to set parameter values for the proposed algorithm and the second portion, the test set was used for evaluation. The parameters are set based on the training set. For our algorithm over the KDD-99 data set, the detection threshold is varied successively and at each threshold, corresponding detection rate and false positive rate are computed. Using this observation, a ROC curve is obtained.

The parameter settings are as follows. For the nCP cluster-based algorithm with canopies presented in this thesis, the cluster width of the fixed-radius clustering was set to be 40. This value will differ depending on the size of the dataset and the chosen n value.. It is generally around 0.5 percent of the chosen n value. However a reasonable width should be set by testing for various values (around the said 0.5%) to achieve

efficient processing time. Also choosing an inefficient width will not affect the detection rate but the efficiency of the process. The clustering width is set in such a way that most of the data is grouped together into clusters. If the cluster width is too small, then the data is grouped together into many small clusters. If the cluster width is too large, then most of the data is grouped into very few clusters. The n value of the nCP algorithm was set to 10,000 of the data set. The n is adjusted to the overall size of the data and it takes a value that is around 1%-2% of the overall size of the data. This value also does not influence the detection rate of the algorithm.

The experiments were performed on the data set for various findings. The first experiment was conducted by applying the algorithm on all the 41 features and then on the selected 17 features [47] based on the importance of the fields. The detection accuracy was obtained from both the tests. Each test involved training and testing phases. The experiment with 17 features included the data reduction phase. In the data reduction phase, important variables for intrusion detection were selected as described in [47] and then the training and testing phases were conducted. The selected 17 features are listed in the table 2.

The second experiment analyzes the performance measure of both the Y-Means algorithm and the nCP clustering algorithm with canopies technique. Both algorithms are tested for their detection accuracy and false alarm rate to depict the advantages and accuracy of my proposed algorithm. Here we compare our algorithm with Y-means because both the algorithms are devised to avoid the shortcomings of the K-means and strive to outperform the K-means algorithm, the basis concept for both the algorithms compared here.

duration
protocol-type
service
src_bytes
land
wrong_fragment
num_failed_logins
logged_in
root_shell
num_file_creations
is_guest_login
count
srv_count
serror_rate
srv_serror_rate
diff_srv_rate
dst_host_count

Table 2: Selected important 17 features out of 41 features in KDD-99 data set

The third and important experiment involves the analysis of time efficiency of the proposed clustering algorithm with canopies technique to show the speed up of the algorithm with the use of canopies compared to the basic K-Means and Random K-Means algorithm. The random K-means algorithm differs from the basic K-means only in the way; the centroids are placed in the first step. Basic K-means plots first n points as the centroids and starts the iteration, whereas in random K-means clustering, n centroids are chosen at random and then proceeds with the grouping iteration. This test would show the time efficiency of the nCP clustering algorithm integrated with canopies concept. The basis of this algorithm is K-means algorithm and we try to eliminate the shortcomings of the K-means algorithm and make it more efficient than the K-means does. The efficiency

is always measured and compared with the algorithm from which we derive or to which we apply some optimization technique. Here such algorithm is K-means and so we compare it with K-means algorithm.

6.3 Experimental Results

The approach of clustering technique integrated with the concept of canopies performed well over the KDD-99 data set.

The results of the first experiment performed on KDD-99 data set with varying number of network connection records features which in this case 41 and 17 are shown in the table 3. The training and test comprises of 5092 and 6890 records. The detection accuracy values clearly depict the increase in the accuracy level for each of the attack class with maximum of 0.94% increase for the normal data values. This observation can be used to increase the efficiency of the clustering algorithms that work on KDD-99 data set.

	41 Variables			**17 Variables**		
Attack Class	Train(s)	Test(s)	Accuracy (%)	Train(s)	Test(s)	Accuracy (%)
Normal	98.17	61.42	88.79	57.28	39.03	89.73
Probe	117.12	69.07	87.22	61.56	32.47	88.06
DoS	128.28	73.22	87.54	68.16	35.28	87.81
U2R	72.45	40.39	57.45	37.48	18.54	57.61
R2L	106.21	52.51	88.12	55.42	39.31	88.62

Table 3: Performance of nCP clustering Algorithm with Canopies

The performance of the nCP clustering algorithm is shown using the ROC curve in the figure 6.1. The ROC curve of the nCP clustering shows the maximum detection accuracy of 89% and false positive accuracy of 8%. The Y-Means clustering algorithm gives the maximum detection accuracy of 86% and false positive accuracy of 2.72%. On analyzing the ROC curve, the nCP clustering shows the higher detection rate than Y-Means clustering algorithm at all selected points. However Y-Means clustering algorithm has better false positive rate at all threshold points. Based on this observation it can be concluded that nCP clustering with canopies algorithm has a better detection rate than Y-Means clustering and the application that demands better false alarm rate can apply the Y-Means clustering technique for intrusion detection.

Algorithm	**Detection Rate**	**False Positive Rate**
nCP Cluster	89.71%	8.32%
nCP Cluster	77.12%	6.93%
nCP Cluster	54.23%	6.31%
nCP Cluster	23.42%	4.17%
nCP Cluster	11.03%	3.21%
Y-Means	86.63%	2.72%
Y-Means	74.38%	2.12%
Y-Means	62.12%	1.81%
Y-Means	21.37%	1.27%
Y-Means	10.67%	1.15%

Table 4: Selected points from the ROC curves of the performance of each algorithm over the KDD Cup 1999 Data

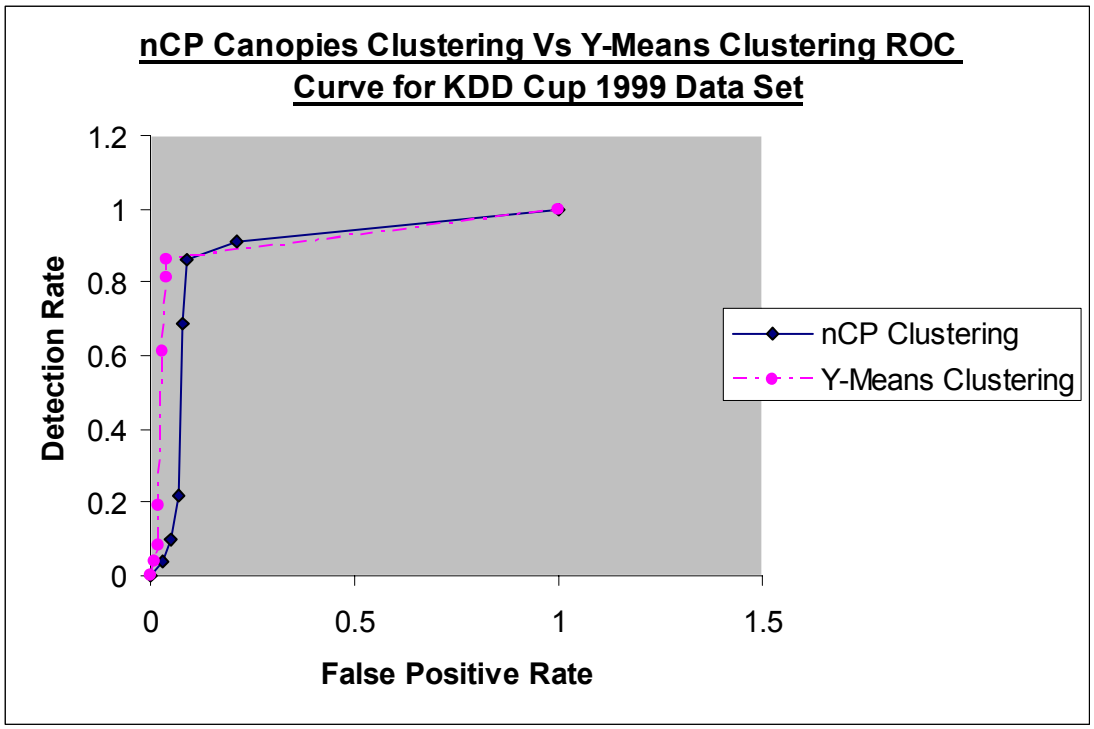

Figure 6: ROC performance curves for nCP and Y-Means Clustering

The third experiment is performed on the KDD Cup 1999 data set to determine the time efficiency of the nCP clustering algorithm compared to basic K-Means and random K-Means Clustering. Time is computed from the start of the algorithm to the stage that forms complete clusters of all the data records from the KDD Cup 1999 data set. Total records of 19,514 records are passed to each algorithm and computing time is measured till the complete cluster formation using all the input data records. Table 5 shows the results of this experiment.

Clustering Algorithm	Total Records	Time Consumed
K-Means	19,514	314.12
Random K-Means	19.514	349.27
nCP with Canopies	19,514	248.34

Table 5: Time consumed by each algorithm on KDD-99 data set for cluster formation

The nCP clustering shows the higher time efficiency of 248.34 seconds with the use of canopies clustering technique compared to K-Means and random K-Means clustering. This observation leads to the conclusion that canopies technique speeds up the process of cluster formation. K-Means clustering takes next position with the time consumption of 314.12 seconds and random K-Means clustering consuming 349.27 seconds for total of 19,514 data records.

The fixed width algorithm takes the time complexity of O(n), where n denotes the total number of data points. This algorithm can plot all the data points in single pass and the running time is directly proportional to the total number of data points available for clustering. The n-cp clustering also has the time complexity of O(n), where n represents the total number of clusters available to break and grab the required n closest data points that are necessary to compute the nCP score.

CHAPTER VII

CONCLUSION

We have presented a clustering framework to detect intrusion and an algorithm using the canopies concept to speed up the detection process. The data points are plotted in a complete metric space to provide the mathematical ease in distance computation. The canopies technique is efficiently used to form the complete clustering framework. A good training dataset will provide a more orderly and superior basis for the algorithm to work on the intrusions. The substructures called canopies appear also in an overlapping fashion that forces the cluster gap to be well defined. The cluster gap is seriously considered and recomputed on each iteration to speed up the process and produce more accurate results. The algorithm was tested against the KDD Cup 1999 data set. The experimental results show relatively high detection rate and low false alarm rate.

Future work in this project includes the improvisation in false alarm rate. The proposed algorithm has a good false alarm rate but it is relatively higher when compared to the Y-means algorithm. More accurate distance computation using a complex and more reliable formula should bring a better false alarm rate. However this would increase its complexity and the time efficiency.

REFERENCES

[1] Stephen Northcutt, Judy Novak, Network Intrusion Detection: An Analyst's Handbook (2nd Edition), Sams publishing, 2000.

[2] http://netsecurity.about.com/cs/hackertools/a/aa030504.htm (Last Referenced: January 14, 2005)

[3] Tim Crothers, Implementing Intrusion Detection Systems: A Hands-On Guide for Securing the Network, Wiley Publishing, Inc., 2002.

[4] Mark Cooper, Stephen Northcutt, Matt Fearnow, Karen Frederick, Intrusion Signatures and Analysis, Sams publishing, 2001.

[5] http://www.windowsecurity.com/articles/Hids_vs_Nids_Part1.html (January 14, 2005)

[6] http://www.sans.org/resources/idfaq/what_is_id.php (January 14, 2005)

[7] http://www.sans.org/resources/idfaq/host_based.php (January 14, 2005)

[8] http://www.sans.org/resources/idfaq/network_based.php (January 14, 2005)

[9]http://www.sans.org/resources/idfaq/aint.php, http://www.nwfusion.com/reviews/2005/ 013105rev.html (January 14, 2005)

[10] Breunig M., Kriegel H., Ng R. T., Sander, "LOF: Identifying Density-Based Local Outliers", Proceedings of the ACM SIGMOD Conference, pp. 24-31, 2000.

[11] Aggarwal C. C., Yu P S., "Outlier Detection for High Dimensional Data", Proceedings of the ACM SIGMODConference, pp. 45-49, 2001.

[12] Ramaswamy S., Rastogi R., Shim K., "Efficient Algorithms for Mining Outliers from Large Data Sets", Proceedings of the ACM SIGMOD Conference, pp. 13-21, 2000.

[13] Knorr E M., Ng R. T., "Algorithms for Mining Distance-Based Outliers in Large Datasets", Proceedings of the VLDB Conference, pp. 78-85, 1998.

[14] Sequeira, K., Zaki, M.: Admit: Anomaly-based data mining for intrusions. In: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, ACM Press, pp.386–395, 2000.

[15] Leonid Portnoy. "Intrusion detection with unlabeled data using clustering", Undergraduate Thesis, Columbia University, 2000

[16] Wenke Lee K., Salvator J. Stolfo, "Data Mining Approaches for Intrusion Detection", Proc. 1998. 7th USENIX Security Symposium, 1998.

[17] William W. Cohen, "Fast Effective rule induction", Proceedings of the 12th international Conference on Machine Learning, 1995.

[18] William W. Cohen, "Text categorization and relational learning", Proceedings of the 12the International Conference on Machine Learning, pp. 112-118, 1995.

[19] Wenke Lee K., Salvator J. Stolfo, Mok K., "Mining Audit Data to build Intrusion Detection Models", pp. 56-64, Proceedings of KDD-98, 1998.

[20] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, and Pang-Nig Tan, "Data Mining for Network Intrusion Detection", Next Generation Data Mining, pp. 23-34, 2002.

[21] John E. Dickerson, Jukka Juslin, Ourania Loulousoula, Julie A. Dickerson, "Fuzzy Intrusion Detection", IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference, pp. 85-90, 2001.

[22] Bezdek J.C., Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, 1981.

[23] Price, K. E., "Host-based misuse detection and conventional operating systems' audit data collection", M.S. Thesis, Purdue University, 1997.

[24] Ptacek, T. H. and Newsham, T. N., "Insertion, evasion, and denial of service: Eluding network intrusion detection", Tech. Rep., Secure Networks, Inc., 1998.

[25] Ilung, K., Ustat K.M., "A real-time intrusion detection system for UNIX", IEEE Symposium on Security and Privacy, Oakland, CA, pp. 16–28, 1993.

[26] Bellovin, S. M., "Packets found on an Internet", Computer Communications Review pp. 23, 3, 26–31, 1993.

[27] Paxson, Bro V., "A system for detecting network intruders in real-time", Computer Networks, pp. 31, 23/24, 2435–2463, 1999.

[28] Kumar, S., “Classification and detection of computer intrusions”, Ph.D. Thesis, Purdue University, 1995.

[29] Lee, W., Stolfo, S. J., “A framework for constructing features and models for intrusion detection systems”, ACM Transactions on Information and System Security, pp. 3, 4, 227–261, 1998.

[30] Mounji, A., “Languages and tools for rule-based distributed intrusion detection”, Ph.D. Thesis, Facultes Universitaires Notre-Dame de la Paix Namur, Belgium, 1997.

[31] Ning, P., Jajodia, S., Wang, X. “Abstraction-based intrusion detection in distributed environments”, ACM Transactions on Information and System Security, pp. 4, 407–452, 2001.

[32] Zamboni, D., “Using internal sensors for computer intrusion detection”, Ph.D. Thesis, Purdue University, 2001.

[33] Almgren, M., Debar, H., AND Dacier,M., “A lightweight tool for detecting web s rver attacks”, Network and Distributed System Security Symposium, pp. 157–170, 2000.

[34] Sekar, R., Guang, Y., Verma, S., Shanbhag, T., “A high-performance network intrusion detection system”, 6th ACM Conference on Computer and Communications Security, pp. 8–17, 1999.

[35] Portnoy, L., Eskin, E., Stolfo, S., “Intrusion detection with unlabeled data using clustering”, ACM Workshop on Data Mining Applied to Security, pp. 231-252, 2001.

[36] Sequeira, K., Zaki, M, “Admit: Anomaly-based data mining for intrusions”, Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, ACM Press, pp. 386–395, 2002.

[37] Guan, Y., Ghorbani, A.A., Belacel, N, “Y-means: A clustering method for intrusion detection, Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, Montreal, Canada, pp. 87-99, 2003.

[38] Munson, J., Wimer, S.: Watcher: the missing piece of the security puzzle. In: Proceedings of the 17th Annual Computer Security Applications Conference, New Orleans, LA, USA, IEEE Computer Society, pp. 230–239, 2001.

[39] Cheeseman Peter, Stutz John, “Bayesian classification (autoclass): Theory and Results”, ACM Transactions on Information and System Security, pp. 153–180, 1996.

[40] Cheeseman Peter, Stutz John, Will Taylor. World Wide Web, http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/ (January 15, 2005)

[41] Yu Guan, Ali A. Ghorbani, Nabil Belacel. World Wide Web, http://www.cs.unb.ca/profs/ghorbani/ali/papers/ccece03.pdf (January 15, 2005)

[42] Nong Y., Xiangyang L., "A Scalable Clustering Technique for Intrusion Signature Recognition", Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, pp. 5-6 June, 2001.

[43] Jens Tölle, Oliver Niggemann. Supporting Intrusion Detection by Graph Clustering and Graph Drawing, World Wide Web, http://dbvis.fmi.uni-konstanz.de/members/panse/seminar_ws0203/pdf/SupportingIntrusionDetectionbyGraphClusteringandGraphDrawing.pdf

[44] Peter Mell, Vincent Hu, Richard Lippmann, Josh Haines, Marc Zissman "An Overview of Issues in Testing Intrusion Detection Systems"

[45] Andrew McCallum, Kamal Nigam, Lyle Ungar, "Knowledge Discovery and Data Mining: Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching.", Proceedings of International Conference on Machine Learning", pp. 231-286, 2000.

[46] Luke, Brian T., "K-Means Clustering", http://fconyx.ncifcrf.gov/~lukeb/kmeans.html (November 20, 2005)

[47] Chebrolu S., Abraham A., Thomas Johnson P., "Feature deduction and ensemble design of intrusion detection systems", Computers and Security, pp. 295-307, 2005.

VITA

Sivanadiyan Sabari Kannan

Candidate for the Degree of

Master of Science

Thesis: Y-MEANS CLUSTERING Vs N-CP CLUSTERING WITH CANOPIES FOR INTRUSION DETECTION.

Major Field: Computer Science

Biographical:

Personal Data: Born in Pollachi, Tamil Nadu, India, on April 12, 1980, the son of Mr. S. Kannan and Mrs. Visalakshi Kannan.

Education: Received Bachelor of Engineering in computer Science and Engineering from Madras University, Chennai, India in May 2002.Completed the requirements for the Master of Science Degree with a major in Computer Science at Oklahoma State University in December 2005.

Experience: August 2003 – December 2004: Supervisor of Kerr-Drummond dining hall of Oklahoma State University, Stillwater. Internship offer from StatSoft, Inc. Tulsa from September 2005 to February 2006.

.