**AFRL-IF-RS-TR-2004-338**
**Final Technical Report**
**December 2004**

# MITIGATING THE INSIDER THREAT WITH HIGH-DIMENSIONAL ANOMALY DETECTION

**Telcordia Technologies**

**Sponsored by**
**Defense Advanced Research Projects Agency**
**DARPA Order No. N125**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.**

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

## STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-338 has been reviewed and is approved for publication.

APPROVED: /s/
THOMAS M. BLAKE
Project Engineer

FOR THE DIRECTOR: /s/
WARREN H. DEBANY, JR.
Technical Director
Information Grid Division
Information Directorate

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 074-0188*

| | | |
|---|---|---|
| **1. AGENCY USE ONLY (Leave blank)** | **2. REPORT DATE**<br>DECEMBER 2004 | **3. REPORT TYPE AND DATES COVERED**<br>**FINAL Aug 02 – Aug 03** |

**4. TITLE AND SUBTITLE**

MITIGATING THE INSIDER THREAT WITH HIGH-DIMENSIONAL ANOMALY DETECTION

**5. FUNDING NUMBERS**

C - F30602-02-C-0115
PE - 63760E
PR - N125
TA - 96
WU - 10

**6. AUTHOR(S)**

S. Pramanick
S. Rajagopalan
Eric van den Berg

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Telcordia Technologies
445 South Street
Morristown NJ 07960-6438

**8. PERFORMING ORGANIZATION REPORT NUMBER**

N/A

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFGB
525 Brooks Road
Rome NY 13441-4505

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

AFRL-IF-RS-TR-2004-338

**11. SUPPLEMENTARY NOTES**

AFRL Project Engineer: Thomas M. Blake/IFGB/(315) 330-1482 Thomas.Blake @rl.af.mil

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**12b. DISTRIBUTION CODE**

**13. ABSTRACT *(Maximum 200 Words)***

In this project, we explored new techniques for detecting the threat of insider attacks in enterprise networks. In particular, we explored the use of high-dimensional search techniques such as Latent Semantic Indexing to mitigate the problem of high dimensionality that is inherent in intrusion detection. This new technique can be used for both labeled and unlabeled detection, and shows promise for detecting attacks and anomalies earlier than previously possible and detecting attacks that are similar to past ones.

**14. SUBJECT TERMS**

Intrusion Detection, Anomaly Detection, Semantic Analysis

**15. NUMBER OF PAGES** 31

**16. PRICE CODE**

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT**<br>UNCLASSIFIED | **18. SECURITY CLASSIFICATION OF THIS PAGE**<br>UNCLASSIFIED | **19. SECURITY CLASSIFICATION OF ABSTRACT**<br>UNCLASSIFIED | **20. LIMITATION OF ABSTRACT**<br>UL |

# Table of Contents

# List of Figures

# List of Tables

## 1 Introduction

In the March 2003, NRRC workshop on Cyber Indications and Warnings, the basic question was asked: How can we detect new large-scale attacks as early as possible? Early attack detection or Cyber Indications as it is also known, is not a new topic. However, it is not a well-solved problem either. Every year the world loses billions of dollars in lost data, services, and productivity due to cyber attacks that ravage the network infrastructure. Indeed, all indications are that attacks are rising in scale, effectiveness, and frequency. Setting aside the question of attack prevention, one may simply ask a basic question: what are these "attacks," how do we recognize them, and what would constitute a robust response to a specific attack. We address some aspects of these questions in this paper.

Any cursory analysis of attacks and responses would reveal that we are in an "arms race" with defenders and attackers almost evenly matched. Almost as quickly as new defenses are put up, attacks apparently get modified to defeat them. While we may not be able to eliminate the arms race nature of the equation, it is imperative to make a significant quantum jump in defensive mechanisms to make headway against our adversaries. A significant new phenomenon that has complicated the problem is the emergence of "multi-dimensional" attacks. "Code Red" and other such worms, in effect, exploited vulnerabilities in multiple technologies together to penetrate systems. This has made the problem more complex because of the combinatorial explosion of possible combinations of technologies. Surprisingly, the White House Report on Cyber Infrastructure Protection of 2002 explicitly mentions the multi-dimensional attack problem in the list of top priority problems for the national infrastructure.

A particular kind of attack that is of considerable interest today is the "Insider" attack. The general notion of Insider attacks as those that originate inside the network (as opposed to the Internet or other extranet) is not usable for two reasons: modern network topologies no longer allow a useful distinction of "inside" and "outside" and origins of attacks are not well-defined because of the wide-spread use of "zombied" machines. Rather, Insider attacks are generally considered to be those that are launched (or launchable) from a position of relative privilege in the existing system (the "insiders") where the privilege may vary from resource access level to simply the ability to put illegitimate packets on the network. What makes this problem especially hard is that we cannot foresee which users are bad or will launch an insider attack, we may have to assume the worst, namely, that any user or group of users may launch an attack. Because of the privileged starting point for insiders, these attacks are in a position to inflict wider damage faster than other attacks. Hence being able to detect an attack as quickly as possible is critical to infrastructure protection.

Another problem in the management of complex systems is the poor understanding of the interdependence between technologies. For example, though we know that DNS forms a significant part of the backbone traffic, our ability to correlate DNS failures and traffic load changes is anecdotal at best. This can be explained, at least partly, by the fact that the various technologies have overlapping effects on each other and causality is hard to determine from correlation. Going forward this problem will continue to get worse and we have to find ways of reducing the complexity of the problem. This is traditionally done by increasing the amount of monitoring in the system by adding new attributes or increasing the granularity of the monitoring. However adding new attributes increases the complexity of the correlation task. The main contribution of this paper is that we address the problem of "dimension creep" in correlation by reducing the dimensionality of the problem but at the same time not losing the correlations between the various attributes that need to be extracted for analysis. Because of the difficulty of differentiating anomalies from attacks in a general sense we will use the more general term anomaly in this paper with the understanding that whenever possible we categorize an anomaly as an attack based on past history or by explicit assertion.

Finally, we have the problem that attacks change over time. Signature-based analysis has to be made sufficiently abstract and general in order to catch variants of previously known attacks. The challenge here is to be able to detect new kinds of attacks that have not been seen in the past. These new attacks may be able to elude our radar by adopting new signatures, vectors, and strategies. Today's approaches use existing attacks as models of future ones. This paper makes the first step of going past this model. The intuition here is that rather than analyze traffic with *a priori* models of attacks, we let the data define the distributions that correspond to attacks. Since we do not know which attributes the attack will manifest on we have to formulate an approach that takes all or as many as possible into account in the analysis. We still need a precise notion of attack to define the problem and for the purposes of our current work we assume that human annotation of past history will provide the concept of attack by way of example. Our problem then reduces to looking for network phenomena that are statistically "close" to any attack that has been seen. The hope here is that, to take an example, any variant of the Slammer work that uses a different vector will still show the same or similar traffic characteristics *in an abstract sense.* We propose to test the general hypothesis that attacks can be classified into classes where attacks in a class share some abstract pattern of traffic that is not necessarily tied down to port numbers or applications. We also conjecture that a high dimension search for such attack patterns will lead to earlier discovery of these attacks. We do in this in two ways. In the "labeled" approach, we have annotated states of the network that are used to compare with the current state. The current state is compared with the history of past network states and is labeled with the annotation of the nearest state using some distance metric. The intuition is that in a high dimensional space it is hard for a human to compare multiple attributes at the same time. By providing the comparison metric, the algorithm classifies the current state into various classes of choice, examples could be, "network instability", "worm attack" etc. The second option is unlabeled classification wherein our algorithm merely finds the historical states of the system that are closest but without classifying them into particular good or bad states. It will then be up to a human to label the current state based on the query results. We note that in both cases the objective is to bring to bear the "experience" of the network in categorizing large attribute sets.

The attendant question with attack detection is generating the appropriate response. We have not addressed the question of response in this paper due to space limitations. Suffice it to say that high-dimensional methods of detection enable sophisticated response techniques based on history because our query results also provide comparisons of the current state with similar states in the past and if the responses of the past are also stored they can provide guidelines for the new response.

## 2 Motivation

Quick and accurate identification of anomalies is of great importance in large, complex computer networks. Anomalies may indicate operational problems such as congestion or element faults. Alternatively, they might signal security problems such as an intrusion or a network attack. For both these classes of problems, rapid detection is critical, and automated anomaly detection tools greatly facilitate network management.

In this paper, we focus on the particular application of intrusion detection. Current intrusion detection systems are often signature-based, which allows them to detect known attacks quickly, but they are defenseless against novel attacks. Anomaly detection systems have the capability of detecting previously unknown intrusions. They work by learning ‘normal behavior’, and flag deviations from such ‘normal’ behavior.

In recent years several researchers have applied data mining techniques in systems to find anomalies in (multidimensional) network data. These systems try to construct or ‘learn’ classifiers to distinguish ‘normal’ and ‘anomalous’ data. They need a training data set to practice the classifier. Often, the training set events are assumed to be correctly labeled as either normal or attack. Constructing such a labeled training set is often a laborious and expensive task. Therefore, interest has increased in ‘unsupervised’ anomaly detection of unlabeled data. See for instance [1,2]. This type of anomaly detection relies on the assumption that anomalous elements are very rare compared to normal elements, and exhibit some features that clearly distinguish them from normal elements. It is related to the problem of finding ‘change points’ in single dimensional data, or to finding outliers and clustering in multivariate data.

Depending on the type of data that an Intrusion Detection System uses as input, it is either classified as host-based, or network-based. Typically, a host-based IDS uses data such as audit trails and system logs. A network-based IDS analyzes traces of captured network packets. Data mining techniques can in principle be used for anomaly detection in both types of IDS. For instance, they can operate on connection records abstracted from the captured packet headers, or on (sub) sequences of system calls from a system log [1].

Despite its power and generality, there are several challenges for data mining-based intrusion detection. To mention a few: modeling temporal data (detecting trends, frequency analysis), scalability (efficiency problems for large, high-dimensional feature spaces), and incremental mining (smoothly incorporating new information, without having to reanalyze the entire dataset from scratch).

In the current paper, we propose an anomaly detection system that overcomes many of the problems mentioned above. The system is based on Latent Semantic Analysis, a data-mining technique, which has been shown to be very effective for retrieving information in high-dimensional spaces. Its scalability and options for incremental mining are very well studied. Furthermore, the system incorporates detectors for individual network features, to better detect temporal trends.

Latent Semantic Analysis is a data-driven technique, based on Singular Value Decomposition of an appropriately constructed ‘term-document’ matrix. In this initial work, we show the potential of this technique for use in an anomaly detection system.

This paper is organized as follows: in Section 3, we describe our system architecture, as well as methods for data collection and simplex detection. In Section 4, we describe Latent Semantic Analysis, our data-mining system for detecting anomalies in high dimensional data. In Section 5, we describe the mapping of incoming data to our data-mining system. An example of our system on a small database is given in Section 6. Finally, related work is discussed in Section 7, and our conclusions are presented in Section 8.

## 3 Measurement and detection

A typical intrusion detection system (IDS) has three components – *data collection* (and *reduction*), *data classification* and *data reporting*. Since data reporting is normally trivial, in this section, we focus on the means of collecting data regarding the dynamic state of the system and classifying it either as normal or anomalous.

## 3.1 Intrusion Detection Architecture

The generic architecture of our intrusion detection system is illustrated in Figure 1. It introduces a hierarchically layered approach to system surveillance that includes sensors/observables, simplex anomaly and/or misuse detection systems, mapping modules, an alert classifier, a database of observed behavior and a warning module. The sensors are responsible for collecting a target-specific event stream. The event stream may be derived from a variety of sources including audit data, network packets etc. Each event stream is connected to its corresponding anomaly and/or misuse detection system. The detection modules receive large volumes of event logs and produce smaller volumes of intrusion or suspicion reports (alerts) that are then fed to their associated mapping modules. Detailed description of the event streams and their detection modules is given in section 3.2. The mapping module is responsible for normalizing the alerts and converting them into a format understandable by the LSI classifier (see Section 5.2). The LSI classifier enables analysis of diverse alerts and can detect unknown attacks such as Internet worms and coordinated attacks. The known behavior is kept in the attack history database in the form of documents and the observed anomalous behavior is added to the database from time to time. The feedback mechanism is provided so that LSI module can increase the threshold of an anomaly detector if the false positive rate of that detector is too high and reduce it if the false negative is high. It can also be used to ask for more information from the mapping modules.

The novelty of the work lies in combining high-dimensional and diverse alerts together to make more accurate attack detection. To achieve this goal, we obtain a significant reduction in the dimension of the alert space, by projecting these alerts on a suitably chosen subspace. This is done using a Singular Value Decomposition of a particular matrix of alerts by documented network states. The technique is known in information retrieval as Latent Semantic Indexing, or LSI. First, attacks and alerts are converted to pseudo-documents, or queries, containing terms. Similarly, each network state is described in a pseudo-document, in terms of the alerts generated by it. Then, LSI performs data mining on these documents to retrieve a ranked list of documents matching the given query. The framework can be used for both supervised and unsupervised learning. When unsupervised learning is done, the framework will have high rate of false alarms in the beginning but as more and more system behavior get classified as normal the false alarm rates go down and newer and unknown attacks get caught. Alternatively, weights can be assigned to the incoming arcs of LSI so that less importance is given to the detection modules having high rate of false positives. The remaining sections explain these features in greater detail.

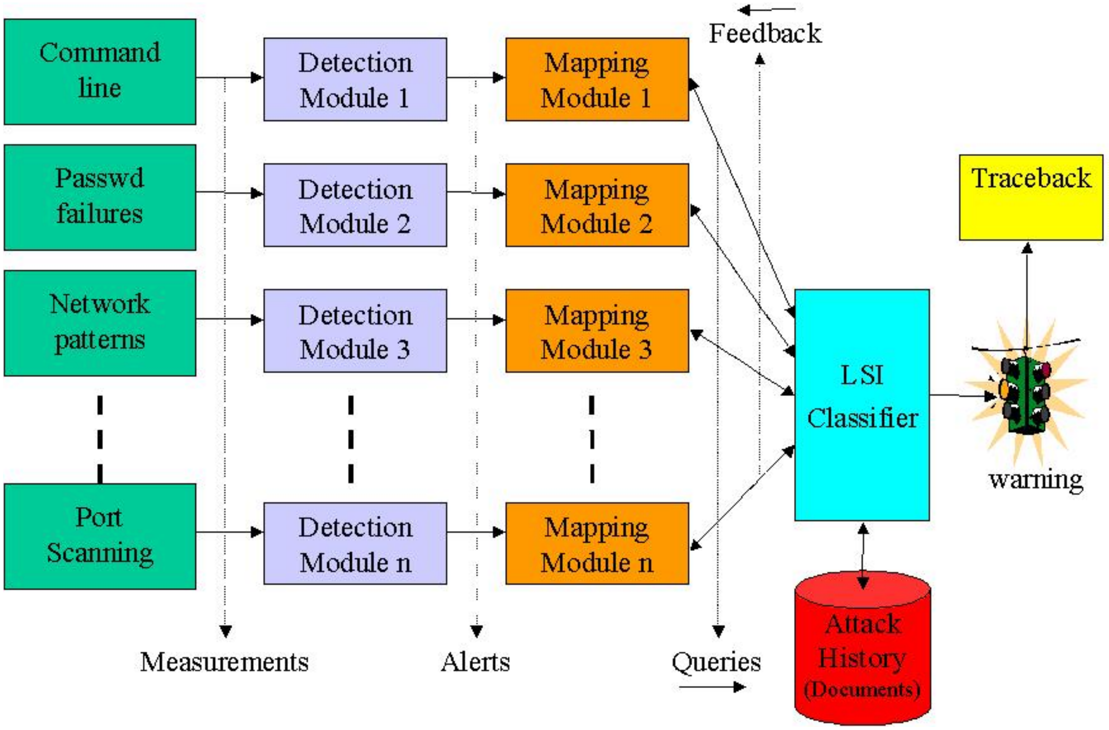

**Figure 1: Detection Framework**

### 3.2 Detection of anomalies in single measurement data

Intrusion detection systems rely on a wide variety of observable data to distinguish between legitimate and illegitimate activities. Most systems collect profiles of user behavior (for example IDES/NIDES [28]), generated by audit logs. Other systems look at network traffic, for example NSM [56]. Other approaches include program specification methods, which attempt to characterize the behavior of privileged processes. Various methods of data collection are described below.

In UNIX systems, system calls provide a natural observable for detecting anomalous behavior, because processes access system resources through the use of system calls. Short sequences of system calls executed by running processes provide a good discrimination between normal and abnormal operating characteristics of several common UNIX programs [49]. System calls can be recorded either by using a tool such as "trace" or by activating the auditing system. In S. Forrest's work [52], [53], a table of characteristic subsequences of system calls is constructed for well-suited processes such as ftpd and sendmail. During real time operation, a pattern-matching algorithm is applied to match on the fly the system calls generated by the process with entries in the pattern table to detect anomalies.

One of the serious threats to the security of computer systems is the masquerade attack, in which one user impersonates another. In the UNIX operating system users execute commands to get their job done. By classifying the sequences of user-command data into either category - self and non-self, the problem of masquerade detection can be addressed. User commands can be captured using the UNIX "acct" auditing mechanism. Schonlau [54], and Maxion [55], have applied various classification techniques on the sequence of truncated user-commands (no arguments) for automatic discovery of masqueraders. By observing the truncated user-commands one can detect insider attacks more thoroughly.

Portscanning is a common activity used by computer attackers to characterize hosts or networks, which they are considering hostile activity against. It is also one of the techniques used by computer worms to multiply themselves. Thus it is useful for system administrators and other network defenders to detect portscans as possible preliminaries to a more serious attack. Various systems such as NSM [56], GrIDS [57], snort [58], Emerald [59], and Spice [60] generate alerts for portscan activities.

Monitoring the server applications provides further insights than the traditional network-based and host-based collection methods, since it can monitor transactions that are encrypted. In [61], an approach for collecting real-time transaction information from a server application is presented that uses a module coupled with the application to extract the desired information. Their experimental results show that the performance impact on the application is tolerable. This application level information can be fed into the anomaly detection unit to get more accurate decision.

Various other common network, host and user measurements are used in anomaly detection. The network measurements include the overall link bandwidth and features of individual connections such as duration, protocol type, number of bytes transferred etc. These measurements can be collected using tools like “tcpdump” an “Iperf” [62]. The host based measurements include the CPU usage parameters, gathered from the “/proc” filesystem, time since last reboot etc. The user measurements include the user profiles such as time of login, duration of user session, cumulative CPU time, names of files accessed, and so forth.

Once the measurements are available, quick detection of potential anomalies is more important than having a high rate of false alarms, as long as evaluating low level alarms is relatively cheap. In Sections 4 and 5 below we describe an efficient method for evaluation of alerts. Fast, online detection can be performed using methods from statistical process control. Methods range from simple direct thresholding to more sophisticated sequential change point detection. Recently, change point detection techniques have been applied to detect SYN flood attacks [43], monitor audit logs [44], and detect DoS attacks [45]. A detailed discussion of statistical change point detection is outside of the scope of this paper. However, such methods aim to minimize the average or maximum detection delay given a fixed false positive rate, or satisfy a related optimality criterion. See e.g. [46]. In general, a test statistic based on the incoming data sequence is incrementally updated, and then compared to a critical threshold. If the threshold is exceeded, an alert is generated.

Separate detectors can be constructed on (possibly transformed) incoming measurement streams, to detect for instance saturation level, rate of increase, individually. In our framework, new aspects of the measurement stream can be monitored by adding a new detector.

To reduce the number of false alarms, as well as increase the number of correct detections, we incorporate a method for aggregating alerts over time. In statistical process control, such aggregations are common in the form of run tests, where an alarm is only generated if the test statistic for the monitored process exceeds a threshold for a given number of times in sequence. Here, we can aggregate alerts over windows of varying (increasing) widths, to collect more and more alerts. The values of alerts can for instance be exponentially smoothed, in order to gracefully de-emphasize old data.

As a side-effect, our usage of alerts functions as a preliminary data filter, as we will see below.

### 3.3 Detection of anomalies in multidimensional measurement data

We can make use of the alerts generated by individual detectors, in order to define our network state in higher dimensions. In general, we can define our network state as a vector of detector values. When an alert is generated by a specific detector, the corresponding vector element is non-zero. Otherwise, i.e. when no alert is generated, the vector element matching a detector is set to zero. Since typically only a few alerts are expected to be generated in a particular network state, this results in a sparse network state description. Note that this approach assumes a potential anomaly generates at least one alert. While this may appear restrictive, we can in fact use two different alert thresholds: the first is purely for the single, individual measurement under consideration. This threshold is set to correspond approximately to a given false alarm rate. A second, lower threshold is used to generate alerts for the higher dimensional analysis described below. In this way, the alerts retain power to signal anomalies for individual measurements, while also retaining flexibility in detecting higher dimensional anomalies where no individual alert is generated (yet). This assumption represents our first cut at this problem, and may be relaxed in later work. As an alternative approach the individual anomaly detectors can supply periodic updates if no alerts are seen within a period.

Further details on the specific values generated by detectors are given in Section 5. In the next section, we describe our method for analysis of network state described by a vector of generated alerts.

## 4 Latent Semantic Analysis

In this section we describe our approach analyzing and classifying network alerts, which is based on Latent Semantic Analysis. Latent semantic analysis or indexing (LSI) has been developed and used successfully for information retrieval [39], finding relevant documents in a database given a query of search terms. This method improves on literal or lexical matching, by solving two problems connected to information retrieval based on queries: synonymy and polysemy. Synonymy broadly refers to the fact that there are many ways to refer to the same object. Polysemy refers to the general fact that most words have more than one distinct meaning.

In the context of anomaly detection, these two problems have the following analogues: first, different combinations of alerts can signal the same form of anomaly. Second, most alerts have different meanings in different (anomalous) network states.

LSI works on a matrix with indexed search terms as rows, and documents as columns. Each entry in the matrix represents the number of times the search term in its row occurs in the document of its column. (In fact, it works on a matrix of weighted counts, where each term and even each entry may be assigned a weight, to indicate relative importance of term or entry).

At the heart of the LSI method is a Singular Value Decomposition (SVD) of the weighted term-document matrix. The SVD is very similar to eigenvalue decomposition, and expresses the original matrix in a combination of linearly independent components or ‘factors’. Most of these factors only contribute little to the overall sum. Therefore, it is possible to reduce the number of factors significantly, without loosing much accuracy. In other words, the original matrix is projected on the smaller, lower dimensional space of most significant factors. All term-term, document-document and document-term similarities are now approximated by their values in this smaller dimensional space.  For information retrieval purposes, SVD can be seen as a method for deriving a set of uncorrelated indexing variables or factors. Each term and document is represented by its vector of factor values. Terms and documents are therefore represented in the same space. Moreover, because the factor space has fewer dimensions, it is possible for documents with somewhat different term usage to be mapped into the same vector of factor values. By replacing the original terms with derived orthogonal factor values, the SVD helps to solve both the synonymy and polysemy problems discussed above. In fact, it deals with synonymy well, but offers only a partial solution for the polysemy problem.

The SVD based approach of LSI has three desirable properties:

1. Adjustable representational richness. By varying the number of factors $k$, we can refine our representation of terms and documents for better precision in retrieval, or coarsen it, to achieve higher data reduction.
2. Explicit representation of both terms and documents. Since terms and documents are represented in the same (factor) space, relevant documents for a search query are simply those, which are close to the query in the factor subspace, by some appropriate distance measure. Moreover, it is relatively simple to add new terms and documents to the factor space, as we will see below.
3. Computational tractability for large datasets. Since the number of factors $k$ is much smaller than the number of terms plus documents $N$, the computational complexity of this method compares favorably to standard vector space based methods. Still, computing the SVD is expensive.

## 4.1 Summary of LSI/SVD approach for anomaly detection

As mentioned in Section 3.3, in our model, network state is described by a vector of detector alert values. In this paper, we stay close to the text based LSI methodology. First, we map (normalized) individual detector values to a semantic description. This is done via ‘qualitative quantization’, by mapping numerical values to words such as ‘high’ ‘medium’ or ‘low’. The detector name is added to the semantic description. Our network state description now consists of a vector of all possible detector-alert terms. When a particular alert is generated by a detector, the vector-element corresponding to the detector alert term is non-zero. When no alert is generated, the corresponding vector element is set to 0. The vector of alert-values so generated is like a query for a search engine. The query is compared to various ‘documents’ consisting of known network state descriptions (in the labeled case), or of previous alerts (in the unlabeled case).

Most search engines perform lexical matching, looking for exact matches with search terms only. Instead, Latent Semantic Indexing first performs a Singular Value Decomposition of the matrix of known ‘search terms’ (the set of detector-alert values) and ‘documented’ network states, consisting, say, of counts of a particular search term (generated alert) in a particular documented network state description (labeled), or previous alert vector (unlabeled). We then construct a subspace spanned by the $k$ most significant factors of this matrix. A new query is projected onto this subspace.  The idea behind LSI is that similar network states and queries, including non-exact, ‘higher-order’ matches, get mapped to the same area of the subspace. Retrieval of related documented network states now consists of finding documented network states close to the query in the subspace. Closeness or similarity can be measured by various metrics. Since LSA operates on a vector space, a popular measure of similarity is the inner product between the projected query vector and potential projected document vectors. Other measures are possible, such as the distance between two vectors in the subspace. Returned documents typically satisfy a distance threshold, like a minimum inner product threshold. If not enough documents satisfy this minimum inner product threshold, then the query is flagged as an anomaly. Note that the projection of our query on the subspace results in a reduction in dimensionality, but also in some information loss. By choosing the basis of our subspace in an optimal way using the SVD, this information loss is minimized for the given document and term vectors. However, if for a particular query, the amount of information lost, as measured by the difference in vector norm between the original query and its subspace projection, exceeds a given threshold, then the query is also designated an anomaly.

### 4.1.1 Labeled Anomaly Detection

The previous description of anomalous queries applies to unsupervised anomaly detection. If the previous 'documented' network states have instead been classified (off-line) according to their anomaly-type (e.g. normal, DoS attack, worm attack, insider attack, etc), then we can return more specific information. In this case, the returned 'retrieved document list' is a sequence of likely states of the network. We can use this list of documented network states together with their likelihood (e.g. as measured by the similarity metric) to further investigate the alert query. When an anomaly is classified as either normal, or some new attack, it can be added to the list of known network states, or 'documents'. This can be done in various ways, from fast and not very accurate 'folding in', via SVD-updating, to full recomputation of the SVD. Labeled anomaly detection allows us to detect attacks in high-dimensional space with more confidence. If the network has encountered attacks that have been annotated correctly, new attacks that are similar but not necessarily identical are now possible to detect using our technique.

In Appendix D, we describe the LSI method in more detail based on [40].

# 5 Network state description

## 5.1 Mapping Attacks to Documents

Different types of attacks such as worm or virus attacks, denial of service attacks, intrusion attacks and insider threats have surfaced in the past decade. These attacks lead the network into some bad state causing unavailability of services, network congestion, reduced quality of service, leak of confidential information and so on.

A Common Intrusion Specification Language (CISL) provides terms for describing an attack in terms of the time, source, target, observer, outcome, attack identity (in terms of a list of well-known attacks and attack classes), and cause [51]. Appendix B shows a typical CISL document for the slammer worm attack. In this work we extend the CISL specifics to represent attacks as documents. We call the new language *Extended Common Intrusion Specification Language* (XCISL). We chose to extend CISL because it is used by the intrusion detection systems to share information about the attacks among themselves. Three new Semantic IDentifiers (SID) are added to XCISL – the *system state*, the *network state* and the *application state*. The system state captures features such as the CPU usage, the system call sequence variation from the norm etc. during the attack. The network state captures the bandwidth usage, the rate of increase in bandwidth demand etc. The application state captures the application in demand during the attack. All the observed intrusive system state are represented as a XCISL document and stored in the attack history database.

Attacks are identified and classified by their attributes, which include the following:

- *Observer/Sensor*: The entity making the attack diagnosis.
- *Initiator*: The entity responsible for carrying out the attack.
- *Target*: The entity that is the target of the attack.
- *Attack Description*: The high level information about the attack. Here we can describe the attack signature, the events that led to the attack, the consequences of the attack and network state when the attack occurred.

We consider web attacks to give an example of a XCISL document. A web attack has a particular *attack life cycle* – entry point, vulnerability, threat, action, input length, target, scope and privileges. The attack life cycle describes a succession of steps followed by an attacker to carry out some malicious activity on the web server. For example, buffer overflow vulnerability in ISAPI extension (idq.dll) in Index Server 2.0 and Indexing Service 2000 in IIS 6.0 beta and earlier allows remote attackers to execute arbitrary commands via a long argument to Internet Data Administration (.ida) and Internet Data Query (.idq) files [50]. The attacker first probes the network for the presence of an IIS web server having the above vulnerability. This is followed by exploitation of the vulnerability by an action, using an HTTP element of certain length. Thus the attacker is able to execute arbitrary commands (deleting, overwriting etc.) at the system level on the web server.

The web attack exhibits anomaly at various levels of the network stack. Probing the subnet for vulnerable web server might trigger off the network based IDS, receiving a message of a particular length and signature might trigger off the host based misuse detection system and executing arbitrary commands that violate the system call sequence norm can trigger off the host based anomaly detection systems. All these events can be collected along with the observations of the IDSs to represent a particular web attack. Figure 2 shows the XCISL document of a web attack [50]. The actual attributes of the attack are fictitious. In the example, the attack was detected by a misuse-based IDS which observed that the system call sequence of the webserver_software was above the desired threshold. The rest of the document is self-explanatory.

## 5.2 Mapping Alerts to Queries

The alerts generated by the individual event-based detection modules are measured under different dimensions and time scales. In order to perform anomaly detection in multidimensional measurement data these alerts need to be combined. To facilitate this operation, mapping modules are added after the detection modules in the hierarchy (see Figure 1). The mapping modules receive the alerts from the detection modules and generate a pseudo-document of XCISL terms called the “query”. The XCISL query is then fed to LSI for data mining purposes. Figure 3 illustrates the working of the detection module and mapping module in tandem.

The mapping module performs four major operations on the local alerts and filtered audit received from the detection module. It first normalizes the parameter-values of the input. This is important because a server with a capacity of 100 connections per second is overloaded when more than 100 connections arrive in second but a server with a capacity of 10 connections per second is overloaded when just 11 connections are received in a second. The comparisons will be

```
( And
  ( ByMeansOf
    ( Attack
      ( Initiator
        ( IPV4Address remote_user )
      )
      ( Observer
        ( ProcessName intrusion detection_system)
          (Mechanism misuse_detection)
          (Threat authorization_violation)
      )
      ( Target
        ( IPV4Address webserver_software)
      )
      ( AttackSpecifics
        ( Certainty 100 )
        ( Severity 20 )
        ( AttackID httpget_oracle)
      )
      ( Outcome
        ( CIDFReturnCode success )
      )
      ( When
        ( BeginTime Wed Jun 16 09:07:46 2000 EDT )
        ( EndTime Wed Jun 16 09:08:46 2000 EDT )
      )
    )
     ( SystemState
        (CommandSeqChange  10%)
        (SyscallSeqChange   70%)
     )
     ( NetworkState
       ( Initiator
          ( IPV4Address remote_user )
       )
       ( Receiver
         ( TCPPort 80 )
         ( IPV4Address webserver_software )
       )
       (LinkBandwidth LOW)
        (LinkBandwidthRate  +0.2)
        (PortScanRate  LOW)
       ( When
         ( BeginTime Wed Jun 16 08:57:46 2000 EDT )
         ( EndTime Wed Jun 16 09:08:46 2000 EDT )
       )
     )
     (ApplicationState
        (InDemandService oracle)
        (MessageLength  fixed)
     )
  )
)
```

**Figure 2: XCISL document of a web attack**

made simple if the readings are recorded as a percentage of the server capacity rather than the actual values. The same concept holds for other measurements such as bandwidth, deviation of system calls etc. The normalization operation is provided a table of weighing factors and normalization schemes for different measurements. The weighing factors in this table can be changed based on the feedback of the LSI classifier (see Figure 1).

Once the normalization is done we end up with a list of parameter value pairs that are filtered further based on the requirements of XCISL. Filtering is done because the XCISL has a fixed set of keywords with which it creates a document. As more and more keywords are added to XCISL less and less parameters will be filtered. An XCISL keyword table is fed to the filtering unit to help it select the desired parameter value pairs.

The final member of the mapping module is the query generator. It receives the filtered parameter value pairs and outputs a pseudo-document in XCISL terms called the XCISL query.

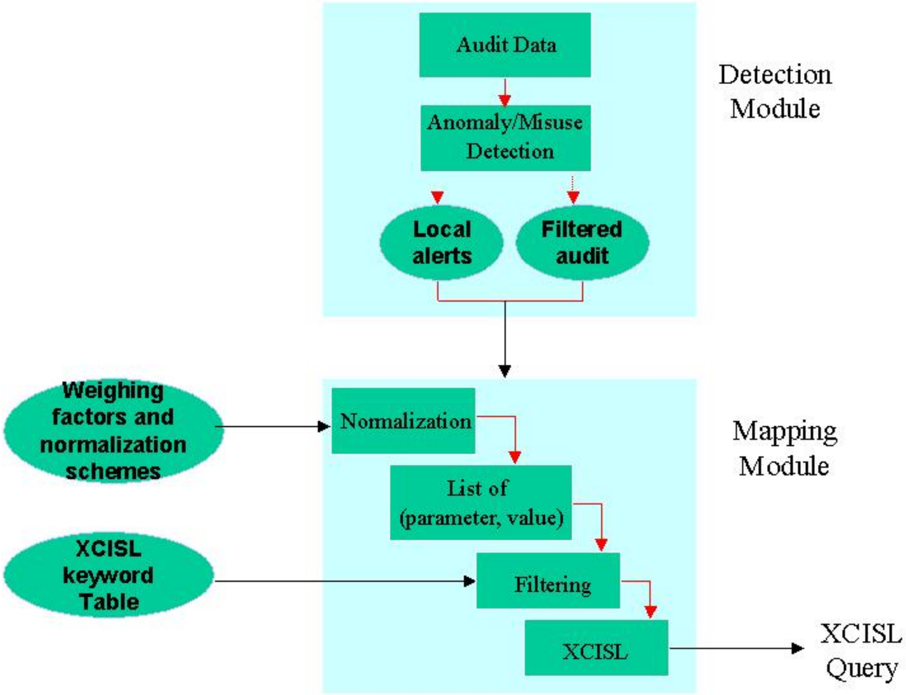

**Figure 3: Encoding system measurements into XCISL style queries**

## 6 Attack Scenario for Evaluation – Description of Queries

In this section, LSI is applied to a small labeled database of attacks and normal behavior. Three kinds of attacks are considered – insider attack, worm attack and denial of service attack. Without loss of generality, for evaluation purposes, we have used normal terms rather than using XCISL keywords for describing system behavior. The XCISL keywords are converted to normal terms by tagging them with their values. For example, (LinkBandwidth LOW) is converted to low_bw, (InDemandService httpd) is written as highdemand_httpd and (CommandSeqChange 10%) is written as 10_commandseq. Since we are using normalized values for the measurements, the number of terms created will be finite. For simplicity of understanding we have used two levels for all the measurements – normal and high usage.

In Table 1, 17 system states are listed. All the italicized terms are used as referents to the system states. In total 16 terms have been identified. Detailed description of the terms is given in Appendix A. The cluster {S1, S2, S4, S6, S8, S10, S13, S14, and S15} belongs to the normal behavior, {S5 and S16} belongs to the DoS attack, {S11 and S12} belong to the insider attack and {S3, S7 and S17} belong to the worm attack. Keeping to LSI terminology the system states are henceforth referred as documents. Corresponding to the measurements in Table 1 is the 16×17 term-document matrix is shown in Table 3 (Appendix C). The elements of this matrix are the frequencies in which a term occurs in a document. The truncated SVD (with $k = 2$) of the 16×17 matrix in Table 3 is shown in Figure 4.

In Figure 4, we find that LSI does a nice job of clustering the system states belonging to the same type close to each other except for S6. Since S6 states that a critical web resource is in heavy demand, a typical starting point of DoS, it got clustered with the DoS attacks even though the system call sequence of the process was normal. As more documents that pay stress on the system call sequence are added, S6 will get clustered in the set of normal behavior.

**Table 1: Database of system state measurements for attacks and normal behavior**

| Label | Measurements |
|---|---|
| S1 | Opnet_simulation; *high_cpu; normal_portscan* |
| S2 | Periodic_update; *normal_portscan* |
| S3 | *High_bw*; *highdev_syscallseq*; *highdev_httpd*; *high_portscan* |
| S4 | Resource_request; *norm_res; norm_bw; norm_portscan* |
| S5 | *High_bw; highdemand_httpd* |
| S6 | *Highdemand_httpd; norm_syscallseq; critical_resource* |
| S7 | *Critical_resource; high_bw; highdev_httpd* |
| S8 | *Norm_commandseq; norm_syscallseq; norm_httpd; norm_portscan; norm_bw* |
| S9 | *Norm_cpu; norm_syscall* |
| S10 | *Norm_httpd; norm_bw; norm_portscan* |
| S11 | Copying_files; *High_passwdattempts; highdev_syscallseq; norm_bw; norm_portscan; highdev_commandseq* |
| S12 | Deleting_files; *High_passwdattempts; highdev_syscallseq; norm_bw; norm_portscan; highdev_commandseq* |
| S13 | *Norm_cpu; norm_res; norm_bw; norm_portscan* |
| S14 | *Norm_commandseq; norm_bw; norm_portscan* |
| S15 | *Norm_bw; norm_portscan* |
| S16 | *High_cpu; critical_resource* |
| S17 | *High_cpu; high_portscan; highdev_syscall* |

Once the SVD is done we wait for a new system state and try to compare it with the previously observed behavior to detect anomalies. The new system state is converted to a query and fed into the LSI. For our evaluation we consider a new worm attack that exhibits system alerts of *high_portscan* and *highdev_syscallseq*, which has not been observed before by the system. Figure 4 shows the mapping of this query to the already formed clusters. All documents whose cosine with the query vector is greater than 0.90 are in the shaded region. The matched documents include {S17, S3, S6, S16, S5 and S7}. A different cosine threshold could be used so that a larger or smaller set of documents is returned. In this example, using a cosine threshold of 0.55 returns S11 and S12 as well (S11 and S12 are somewhat related). With lexical-matching only four documents {S3, S11, S12 and S17} are returned. Hence, the LSI approach can extract four additional documents {S5, S6, S7, S16}, which are relevant to the query yet, share no common terms.

Table 2 lists the LSI-ranked documents with different numbers of factors (*k*). The documents returned in this table are within a cosine-threshold of 0.20 of the query. It is evident that the rank ordering of the documents can vary significantly with changes in the number of factors *k*. As the value of *k* increases more appropriate matches are returned, e.g., S6 which was a normal state and was returned as a match with *k=2* is not present in *k=4* and *k=8*.

## 7 Related Work

The related work in this area is divided into two categories – first, clustering and dimensionality reduction schemes used for data mining and second, intrusion detection schemes using data mining.

*Clustering*

Clustering is a well-known problem and has been studied in many fields including statistics [8], machine learning [9], databases [10], and visualization. Many data-mining algorithms in the literature find outliers as a side-product of clustering algorithms [24, 25, 26, 27, 28, 29]. These techniques define outliers as points that do not lie in clusters, thus implicitly defining outliers as background noise. Another class of techniques [34, 35, 36] defines outliers as points that are neither a part of a cluster nor a part of the background noise; rather, they are points that behave very differently from the norm. But they are not methods that are specifically designed in order to deal with the curse of dimensionality. In [37], the data points are modeled using a stochastic distribution, and points are determined to be outliers depending upon their relationship with this model. However, with increasing dimensionality, it becomes difficult and inaccurate to estimate the multidimensional distributions of the data points. Two algorithms [35], [36] define outliers by using the full dimensional distances of the points from one another. However, in high dimensional space, the data is sparse and the notion of proximity fails to retain its meaningfulness.

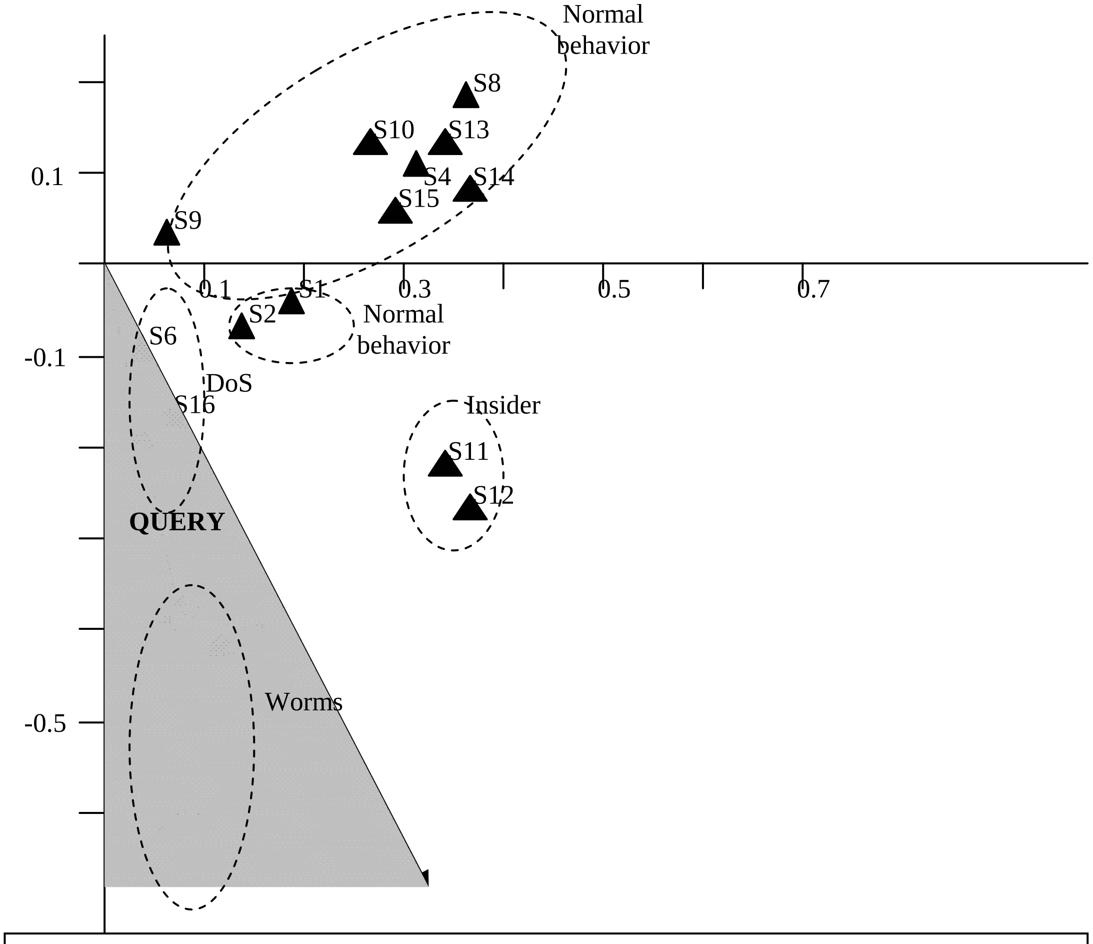

**Figure 4: A Two-dimensional plot of terms and documents along with the query application theory.**

One solution to the problem of high dimensionality is to project the data onto lower dimensions and then compare the data density. The most obvious class is dimensional models, like multidimensional scaling and factor analysis, where representational power can be controlled by choosing the number, *k*, of dimensions (i.e. *k* parameters per object). In [11] two algorithms for outlier detection have been presented for high dimensional problems. The basic approach is the naïve brute force algorithm that examines all possible subsets of *k*-dimensional candidate projection and retains the one that has the most negative sparsity coefficient. The brute force algorithm is very slow at finding the best patterns because of its exhaustive search of the entire space. In the second scheme ideas from a class of evolutionary search methods are used to create an algorithm determining the most effective subset of dimensions for outlier detection. For real time analysis, recomputing the data classes for every outlier detection is expensive, so we work with a fixed set of dimensions. Our scheme, LSI, uses SVD [33] to estimate the original relationships in the high-order structure into *k* linearly independent vectors or factor values (dimensions). The evolutionary algorithm suggested in [11] could be used periodically to update the most effective dimensions on which the high-dimensional data is projected.

*Intrusion Detection*

Our work represents a new approach to anomaly detection in high-dimensional feature spaces. It is interesting to compare this to other data-mining based approaches to anomaly detection, see for instance [1, 2]. In [2], the authors perform intrusion detection with unlabeled data using clustering techniques. In [1], the authors present a general geometric framework for unsupervised anomaly detection, using besides clustering also k-nearest neighbor search and One Class SVM as techniques for detecting intrusions. In both applications, choice of an appropriate feature space and distance metric is crucial to performance. Feature vectors are either labeled ‘normal’ or ‘anomalous’. As mentioned before, a major assumption underlying the labeling is that anomalies are 1) significantly different from normal behavior, and 2) normal instances vastly outnumber anomalies. Several efficiency improvements are suggested, but implementation in a real-time system is challenging.

In many ways, our approach to anomaly detection is similar to the one described above. However, the new LSI/SVD methodology has two main advantages:

- Dimension reduction: After computing the SVD, similarity search is performed in a much smaller dimensional space.
- Addresses synonymy: Different alerts that signal the same attack, may be far apart in feature space, but are close together in the k-factor space. For unlabeled anomaly detection, this will tend to reduce the number of clusters. For labeled anomaly detection, this can indicate possible attacks where none might be suspected otherwise.

Furthermore, the scaling behavior of LSI is well studied, and it has already been implemented in a real-time system for information retrieval [3].

A technique developed at SRI in the Emerald system [22] uses historical records as its normal training data. It then compares distributions of new data to the distributions obtained from those historical records. Discrepancies between distributions signify an intrusion. The problem with this approach, however, is that if the historical distributions contain intrusions the system may not be able to detect similar intrusions in the new instances. Our approach is able to handle noise in the data based on the assumptions that the normal data greatly outnumbers the occurrence of intrusions and anomalies are significantly different from normal behavior. The eBayes [23] system is a newly developed component for the statistical anomaly detection in EMERALD. Given a naïve Bayes model, training data and a set of hypothesis, a conditional probability table is built for the hypothesis and variables, and is adjusted for the current observations. But eBayes may be computationally expensive as the number of hypothesis states increases. By performing SVD on alerts rather than the raw data we overcome the need for heavy computation.

Because anomaly detectors look for abnormalities, many of the data mining techniques that seek to identify outliers in data become readily applicable. In one such scheme [5], the target system is first exercised in an attack free environment and data collected at the process, system and network levels. Principal Component Analysis (PCA) [6], [7] is used to reduce the dimensionality of the collected data. Fuzzy clustering is then performed on the data to obtain clusters that model the normal behavior of the system. Finally, the system is placed under attack and data containing anomalous behavior is collected and compared to the clusters for anomaly detection. Drawback of this approach is that low-level kernel data at the process, system and network levels are collected and this increases the overhead of the clustering algorithms. Furthermore, during the training phase the scheme works on labeled data, which is difficult to obtain. Though our scheme for dimensionality reduction, LSI, is based on PCA it has major improvements over [5]. It uses inner product of feature vectors for recall, which is less time consuming as compared to fuzzy clustering used in [5]. Another improvement of our approach over [5] is that considering alerts rather than low-level system data reduces the input data fed to the classification engine. Also the scheme in [5] uses a subset of 12 dimensions, which can be small for proper classification whereas LSI has been shown to perform well in real-time environment for more than 100 dimensions [3].

Other applications of data mining to anomaly detection include ADAM (Audit Data Analysis and Mining) [18, 19, 20], IDDM (Intrusion Detection using Data Mining) [21] and MINDS (Minnesota Intrusion Detection System) [30, 31]. ADAM (developed at George Mason University for Secure Information Systems) uses a combination of association rules for mining and classification to discover attacks in TCP dump data. It builds a repository of normal frequent classes and attack classes by mining on labeled data. During runtime connections are classified either as normal, known type of attack, unknown type of attack or a false alarm. There are two significant differences between ADAM and our scheme. First, our scheme can handle unlabeled data and second it handles data at higher dimensions than TCP data. IDDM characterizes change between network data description at different times, and produces alarms when detecting large deviations between descriptions. However, IDDM has problems achieving real-time operation. MINDS anomaly detection module assigns a degree of being an outlier to each data point, which is called the local outlier factor (LOF) [32]. Since LOF involves calculating pairwise distances between all data points, which makes it computationally infeasible for millions of data points, MINDS uses a sample training set from the data and compares all data points to this small set which reduces the complexity. The drawback of this approach is that for high-dimensional data it becomes difficult to obtain a training set from the input data, which we overcome by letting the data define the distribution. Moreover, all the above approaches don’t involve any dimensionality reduction so they don’t scale well for high dimensional data.

Data mining techniques have been widely used in misuse based intrusion detection systems as well. Examples are JAM (Java Agents for Metalearning) [12, 13, 14, 15, 16], MADAM ID [16], and Automated Discovery of Concise Predictive Rules for Intrusion Detection [14]. They apply data mining to audit data to compute models that accurately capture behavioral patterns of intrusions and normal activities. Researchers at Iowa State University report on Automated Discovery of Concise Predictive Rules for Intrusion Detection [17]. A genetic algorithm selects feature subsets to reduce the number of observed features while maintaining or improving learning accuracy. Our scheme performs dimensional reduction in an anomaly based intrusion detection system because misuse based systems are ineffective in detecting attacks that have not already been specified.

**Table 2: Returned documents based on different numbers of LSI factors**

| Number of Factors | | | | | |
|---|---|---|---|---|---|
| *k = 2* | | *k = 4* | | *k = 8* | |
| S17 | .99 | S17 | .87 | S17 | .88 |
| S3 | .99 | S3 | .82 | S3 | .78 |
| S6 | .99 | S12 | .57 | S12 | .37 |
| S16 | .99 | S11 | .57 | S11 | .37 |
| S5 | .98 | S16 | .38 | - | - |
| S7 | .98 | S7 | .38 | - | - |
| S12 | .55 | S1 | .35 | - | - |
| S11 | .55 | S5 | .22 | - | - |
| S1 | .38 | - | - | - | - |

## 8 Conclusions

A self-regenerating system can automatically recover all or most of its function or capacity through failures and attacks. In this paper we addressed a fundamental issue that goes to the heart of self-regeneration, namely early and robust anomaly detection. In particular, we focused on the particular task of intrusion and attack detection. We described a system based on evaluating alerts generated by multiple detectors on individual measurement streams. The underlying idea is that search for anomalies in such a higher dimensional space will lead to earlier discovery of attacks and discovery of new attacks that are modifications of old ones. We have also proposed a methodology, based on Singular Value Decomposition of a suitably constructed alert-state matrix, that significantly reduces the dimensionality of the search space, and helps mitigate the ‘curse of dimensionality’. Using the specific method of Latent Semantic Indexing, we showed the potential of this technique for use in a self-regenerating anomaly detection system.

## 9 References

1. E. Eskin et al., “A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data”, Technical Report, Columbia University 2002?
2. L. Portnoy, E. Eskin, S. Stolfo, “Intrusion detection with unlabeled data using clustering”, Proceedings of the ACM CCS Workshop on Data Mining for Security Applications, 2001.
3. C. Chen et al., “Telcordia LSI Engine: Implementation and Scalability Issues”, Proc. RIDE 2001, Heidelberg, Germany.
4. Vipin Kumar, “Data Mining for Network Intrusion Detection: Experience with KDD '99 Cup Data”.
5. H. Shah, J. Undercoffer, and A. Joshi. Fuzzy Clustering for Intrusion Detection.
6. E. Anderson, Z. Bai, C. Bischof, S. Balckford, J. Demmell, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK User’s Guide. Society for Industrial and Applied Mathematics, 1999.
7. G. H. Golub and C. F. V. Loan. Matrix Computations. The Johns Hopkins University Press, 1989.
8. P. Schnell. A method for discovering data-groups. Biometrica, 6:47-48, 1964.
9. R. Rojas. Neural Networks – A systematic introduction. Springer, Berlin, 1996.

10. A. Hinneburg and D. A. Keim. Clustering methods for large databases: From the past to the future. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA. ACM Press, 1999.
11. C. C. Aggarwal and P. Yu. Outlier Detection for High Dimensional Data, Proceedings of the ACM SIGMOD Conference, 1998.
12. W. Lee, S. J. S., and K. W. Mok. Mining audit data to build intrusion detection models. In Proceedings of the International Conference on Knowledge and Data Mining, NY, 1998.
13. W. Lee and S. L. Stolfo. Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, 1998.
14. W. Lee. A data mining framework for constructing features and models for intrusion detection systems. Technical report, Graduate School of Arts and Sciences, Columbia University, 1999.
15. W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In IEEE Symposium on Security and Privacy, 1999.
16. W. Lee, S. J. Stolfo and K. W. Mok. Adaptive Intrusion detection: a data mining approach. Artificial Intelligence Review, 14:533-567, 2000.
17. G. Helmer, J. Wong, V. Honavar, and L. Miller. Automated discovery of concise predictive rules for intrusion detection. Technical Report TR 99-01, Department of Computer Science, Iowa State University, Ames, IA, 1999.
18. N. Wu. Audit Data Analysis and Data Mining. PhD Thesis, George Mason University, Department of Information and Software Engineering, Fairfax VA, 2001.
19. D. Barbara, S. Jajodia, N. Wu and B. Speegle. Mining unexpected rules in network audit trails. Technical report, George Mason University, 1999.
20. D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. In First SIAM Conference on Data Mining, Chicago, IL. Society for Industrial and Applied Mathematics, 2001.
21. T. Abraham. IDDM: Intrusion detection using data mining techniques. Technical Report DSTO-GD-0286, DSTO Electronics and Surveillance Research Laboratory, 2001.
22. P. A. Porras and P. G. Neumann. Emerald: Event monitoring and enabling responses to anomalous live disturbances. In Proceedings of the 20th National Information Systems Security Conference, Baltimore, MD, 1997.
23. A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In Recent Advances in Intrusion Detection, pages 80-93, Toulouse, France, 2000. Springer-Verlag.
24. C. C. Aggarwal et al. Fast Algorithms for Projected Clustering. ACM SIGMOD Conference Proceedings, 1999.
25. C. C. Aggarwal, P. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. ACM SIGMOD Conference Proceedings, 2000.
26. R. Agrawal, T. Gehrke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. ACM SIGMOD Conference Proceedings, 1998.
27. M. Ester, H. P. Kriegel, J. Snader, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD Conference Proceedings, 1996.
28. S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. ACM SIGMOD Conference Proceedings, 1998.
29. T. Zhang, R. Ramakrishnan and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. ACM SIGMOD Conference Proceedings, 1996.
30. Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., Tan, P.: Data Mining for Network Intrusion Detection, Proc. NSF Workshop on Next Generation Data Mining, Baltimore, MD, November 2002.

31. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P., Srivastava, J., Kumar, V., Dokas, P.: The MINDS - Minnesota Intrusion Detection System, accepted for the book "Next Generation Data Mining".
32. Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. Lof: Identifying density based local outliers. In Proceedings of the ACM SIGMOD Conference, Dallas, TX, 2000.
33. Michael W. Berry. Large-scale sparse singular value computations. The International Journal of Supercomputer Applications, 6(1):13-49, 1992.
34. A. Arning, R. Agrawal, and P. Raghavan. A Linear Method for Deviation Detection in Large Databases. KDD Conference Proceedings, 1995.
35. E. Knorr and R. Ng. Algorithms for Mining Distance-based Outliers in Large Data Sets. VLDB Conference Proceedings, September 1998.
36. S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. ACM SIGMOD Conference Proceedings, 2000.
37. V. Barnett and T. Lewis. Outliers in Statistical Data. John Wiley and Sons, NY 1994.
38. K. Julisch, “Data Mining for Intrusion Detection, A Critical Review”, Research Report, IBM Zurich Research Laboratory, November 2002.
39. S. Deerwester et al. “Indexing by Latent Semantic Analysis”, Journal of the Society for Information Science, 41(6), 391-407.
40. M.W. Berry, S.T. Dumais, “Using Linear Algebra for Intelligent Information Retrieval”, SIAM Review, 37(4), 1995, 573-595.
41. C. Chen et al., “Telcordia LSI Engine: Implementation and Scalability Issues”, Proc. RIDE 2001, Heidelberg, Germany.
42. D. Bassu and C. Behrens, “Distributed LSI: Scalable Concept-based Information Retrieval with High Semantic Resolution”, Proc. Text Mining 2003, San Francisco, May 2003.
43. H. Wang, D. Zhang, and K.G. Shin, “Detecting SYN Flooding Attacks”, INFOCOM 2002.
44. N. Ye, S. Vilbert, and Q. Chen, “Computer Intrusion Detection through EWMA for Autocorrelated and Uncorrelated Data”, IEEE Transactions on Reliability, Vol. 52, No. 1, March 2003.
45. R.B. Blazek, H. Kim, B. Rozovskii, and A. Tartakovsky, “A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods”, IEEE Transactions on Systems, Man and Cybernetics, 2002.
46. M. Basseville and I.V. Nikiforov, “Detection of Abrupt Changes: Theory and application”, Prentice Hall, Englewood Cliffs, NJ, 1993.
47. The Spread of the Sapphire/Slammer Worm. http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html
48. Veronique Abily and Mireille Ducasse. Benchmarking a distributed intrusion detection system based on ASAX: Preliminary results.
49. Steven A. Hofmeyr, Stephanie Forrest and Anil Somyaji. Intrusion detection using sequences of system calls.

http://oval.mitre.org/community/forum/archives/2003-01/msg00003.html

50. http://www.isi.edu/gost/cidf/drafts/language.txt.
51. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, “A Sense of Self for Unix Processes”, In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy* (1996).
52. C. Warrender, S. Forrest, B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models”, *1999 IEEE Symposium on security and Privacy* (1999).
53. Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y. (2001), "Computer Intrusion: Detecting Masquerades," Statistical Science, 2001;16(1):58-74.

Maxion, Roy A. and Townsend, Tahlia N. "Masquerade Detection Using Truncated Command Lines." International Conference on Dependable Systems and Networks (DSN-02), pp. 219-228, Washington, D.C. 23-26 June 2002. IEEE Computer Society Press, Los Alamitos, California, 2002.

54. L.T. Heberlein, G.V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, D. Wolber, "A Network Security Monitor," Proc. 1990 Symposium on Research in Security and Privacy, pp. 296-304, May 1990.
55. Staniford-Chen, S. *et al* "GrIDS: A Graph-Based Intrusion Detection System for Large Networks", Proceedings of the 19th National Information Systems Security Conference, Baltimore, 1996.
56. Martin Roesch. "Snort - Lightweight Intrusion Detection for Networks", 13th Systems Administration Conference - LISA '99, November 7-12, 1999, Seattle, Washington, USA.
57. Phillip A. Porras and Alfonso Valdes, "Live Traffic Analysis of TCP/IP Gateways", Internet Society's Networks and Distributed Systems Security Symposium, March, 1998.
58. Staniford, S., J. Hoagland and J. McAlerney. "Practical Automated Detection of Stealthy Portscans", Journal of Computer Security. Vol 10, Issues 1/2, 2002.
59. Magnus Almgren and Ulf Lindqvist, "Application-Integrated Data Collection for Security Monitoring", LNCS. From *Recent Advances in Intrusion Detection (RAID 2001)*. Springer, Davis, California. October, 2001. Pages 22–36.

http://dast.nlanr.net/Projects/Iperf/

## Appendix A

| **Terms** | **Descriptions** |
|---|---|
| High_bw | High bandwidth consumption |
| High_portscan | The port scanning rate is very high |
| Highdev_commandseq | The command line sequence of the user is deviating from the normal profile |
| Norm_bw | The bandwidth consumption is normal |
| Norm_porscan | The port scanning rate is normal |
| Highdev_httpd | The httpd server is deviating from its normal behavior |
| High_cpu | The CPU is in heavy use |
| Highdemand_httpd | The httpd server is in heavy demand |
| Norm_commandseq | The command line sequence of the user is according to his profile |
| Norm_cpu | The CPU usage is normal |
| Norm_httpd | There is a normal demand on httpd server |
| High_passwdattempts | Large number of password attempts is being done |
| Normal_resource | Requests are coming for a non-critical resource |
| Crtitical_rsource | Requests are coming for a very critical resource |
| Norm_syscallseq | The system call sequence is according to the expected behavior |
| Highdev_syscallseq | The system call sequence is deviating from the normal behavior |

**Appendix B**

CISL representation of the slammer worm attack

```
( And
  ( ByMeansOf
    ( Attack
      ( Initiator
        ( IPV4Address 4.22.160.163 )
      )
      ( Observer
        ( ProcessName NetworkRadar )
         (Attributes
            (Mechanism signature-based)
            (Measurement  network-packets)
            (MisusePattern  payload-contents)
         )
      )
      ( Target
        ( IPV4Address 4.22.160.140 )
      )
      ( AttackSpecifics
        ( Certainty 100 )
        ( Severity 50 )
        ( AttackID Slammer_worm)
      )
      ( Outcome
        ( CIDFReturnCode success )
      )
      ( When
        ( BeginTime Wed Jan 16 09:07:46 2003
EDT )
        ( EndTime Wed Jan 16 09:08:46 2003
EDT )
      )
    )
    ( SendMessage
      ( Initiator
        ( UDPPort 28033 )
        ( IPV4Address 4.22.160.163 )
      )
      ( Receiver
        ( UDPPort 1434 )
        ( IPV4Address 4.22.160.140 )
      )
( When
        ( BeginTime Wed Jan 16 08:57:46 2003
EDT )
        ( EndTime Wed Jan 16 09:08:46 2003
EDT )
      )                  )
  )
( Do
    ( TraceMessage
      ( When
        ( BeginTime Wed Jan 16 09:07:46 2003
EDT )
        ( EndTime Wed Jan 16 09:08:46 2003 EDT
)
      )
      ( Initiator
        ( IPV4Address 4.22.160.163 )
      )
      ( Message
        ( ReferTo 0 )
      )
    )
  )
  ( Do
    ( BlockMessage
      ( Message
        ( IPV4Protocol 6 )
        ( UDPDestinationPort 1434 )
        ( SourceIPV4Address 4.22.160.163 )
        ( DestinationIPV4Address 4.22.160.140 )
      )
      ( When
        ( BeginTime Wed Jan 16 09:07:46 2003
EDT )
        ( EndTime Wed Jan 16 09:18:46 2003 EDT
)
      )
    )
  )
)
```

## Appendix C

**Table 3: The 16X17 term-document matrix corresponding to the system states in Table 2.**

| **Terms** | **Documents** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 | S1 |
| High_bw | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| High_portscan | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Highdev_commandseq | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Norm_bw | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Norm_porscan | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Highdev_httpd | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| High_cpu | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Highdemand_httpd | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Norm_commandseq | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Norm_cpu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Norm_httpd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| High_passwdattempts | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Normal_resource | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Crtitical_rsource | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Norm_syscallseq | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Highdev_syscallseq | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

## Appendix D

### D.1 Linear algebra background: Singular Value Decomposition

Suppose we construct an $m \times n$ matrix $A = [a_{ij}]$ with the $m$ available detectors as rows, and $n$ documented network states as columns. Here element $a_{ij}$ denotes the measured (transformed) value corresponding to an alert by detector $i$ in a particular documented network state $j$. In case there was no alert for detector $i$ in network state $j$, then $a_{ij}$ = 0.
The Singular Value Decomposition of $A$, denoted by SVD( $A$ ), is now defined as:

(1) $$A = U\Sigma V^T$$

where $U^T U = V^T V = I_n$, and $\Sigma = diag(\sigma_1,...,\sigma_n), \sigma_i > 0$ for $1 \le i \le r$, $\sigma_i = 0$ for $j \ge r+1$.
The first $r$ columns of the orthogonal matrices $U = [u_1 u_2 ... u_m]$ and $V = [v_1 v_2 ... v_n]$ correspond to the orthonormal eigenvectors associated with the $r$ nonzero eigenvalues $\sigma_1,...,\sigma_r$. The remaining columns of the two matrices form an orthonormal basis for the null-space of $U$ and $V$, respectively.

Two theorems show how the SVD can reveal important information about the structure of a matrix:
Theorem 1: Let the SVD of $A$ be given by equation (1), and $\sigma_1 \ge \sigma_2 \ge ... \ge \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$. Let $R(A)$ and $N(A)$ denote the range and null-space of $A$, respectively. Then,

1. rank( $A$ ) = $r$, $R(A) = span\{u_1,...,u_r\}$ and $N(A) = span\{v_{r+1},...,v_n\}$
2. dyadic decomposition: $A = \sum_{i=1}^{r} u_i \sigma_i v_i^T$.
3. norms: $\|A\|_F^2 = \sigma_1^2 + ... + \sigma_r^2$ and $\|A\|_2^2 = \sigma_1^2$.

Theorem 2:

Let the SVD of matrix $A$ be given by (1) above, and rank( $A$ ) = $r \le p = \min(m,n)$ and define:

(2) $$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T = U_k \Sigma_k V_k^T.$$

Then:

(3) $$\min_{rank(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + ... + \sigma_p^2.$$

Similarly,

(4) $$\min_{rank(B)=k} \|A - B\|_2^2 = \|A - A_k\|_2^2 = \sigma_{k+1}^2.$$

This means: $A_k$ is the 'closest' rank-$k$ matrix to $A$, and thus is the 'best rank-$k$ approximation' to $A$.

## D.2 Latent Semantic indexing

To implement Latent Semantic Indexing, we first need to construct a matrix of terms by documents, as discussed above. For information retrieval, the elements of the term document matrix are the number of occurrences of each term in a particular document, as discussed above.

(5) $$A = [a_{ij}],$$

where $a_{ij}$ is the number of times term $i$ occurs in document $j$. Usually $A$ is a sparse matrix, since not every word occurs in a particular document. In practice, local and global weights can be applied to vary the importance of terms within or between documents. We can write:

(6) $$a_{ij} = L(i, j) \times G(i),$$

where $L(i, j)$ is the local weight for term $i$ in document $j$, and $G(i)$ is the global weight of term $i$. Recall from equation (1) that $A$ can be decomposed in three matrices: the left-singular vectors $U$, the singular values $\Sigma$, and the right-singular vectors $V$. Furthermore, we can find the 'best' rank-$k$ approximation to $A$ by using the singular vectors corresponding to the $k$ largest singular values (see equation (2)). We can summarize the interpretation of SVD components within LSI as follows:

- The rows of $U$ are considered the term vectors.
- The columns of $V^T$ are the document vectors.
- The diagonal of $\Sigma$ are the singular values.
- $m$ is the number of terms.
- $n$ is the number of documents.
- $k$ is the number of factors.
- $r$ is the rank of $A$.

We can think of the truncated SVD of $A_k$ in (2), as capturing most of the underlying information in the term-document matrix $A$. Since the number of dimensions $k$ is much smaller than the number of terms $m$, terms that never co-occur in the same document, but occur in similar documents, will nevertheless get mapped near to each other in the $k$ dimensional space spanned by the factors.

In our present context: alerts, which are never raised in the same documented network state, but nevertheless signal similar network states, will be close together in the subspace.

## D.3 Queries

Given a query, we need to represent as a vector in our $k$-dimensional subspace in order to compare it to documents/network states. If the query is given as a vector of words, then the projection onto this subspace is:

(7) $$\hat{q} = q^T U_k \Sigma_k^{-1},$$

where $q$ is simply the vector of words in the query, multiplied by the appropriate term weights. In other words, the query is located in the $k$-dimensional subspace at the weighted sum of its constituent terms.

### D.3.1 Search for similar documents

Once the mapping of the query to the $k$-dimensional subspace has been accomplished, the query projection can be compared to document vectors to find similar or nearby documents. One measure of similarity is given by the inner product (cosine) between the query vector and a document vector. Typically, the documents are searched linearly to find nearby documents. The resulting $z$ closest documents, or all documents exceeding some inner product threshold are returned to the user.

### D.3.2 Anomalous queries

In the context of anomaly detection, special attention is paid to the case where no documents/network states satisfy the similarity threshold for a given alert query. Hence, the returned list of network states is empty. In this case, the query is flagged as anomalous.
Similarly, a query is classified as an anomaly when not enough documents are returned.

However, this is not the only possibility for an anomalous query. During normal network operation, relatively few alerts will be generated, resulting in (projected) network states with small norm. However, it is also possible, especially in some previously unknown network state, that a combination of alerts is generated that is 'ignored' by the projection on our $k$-dimensional subspace of most important factors. In other words, this combination of alerts $\tilde{q}$ is mostly in the null-space of $\Sigma_k^{-1} U_k^T$. In this case, the norm of the projection of $\tilde{q}$ is much less than the norm of the original $\tilde{q}$. Hence, we can also signal an anomalous query when the norm of its projection does not exceed a minimum percentage of the norm of the original query.

## D.4 Updating

Suppose we have already performed the SVD of a term-document matrix. If more terms and documents must be added (because of the installation of new detectors, or documentation of additional network states), then two alternatives for incorporating them are possible: re-computing the SVD of a new term-document matrix, or 'folding-in' of the new terms and documents. Re-computing the SVD of a larger term-document matrix results in the most accurate subspace projection, but it is quite expensive. Alternatively, we can project the new terms and document in the current factor subspace, at the cost of some loss of accuracy. This 'folding-in' of new terms and documents is accomplished by computing:

(8) $$\hat{d} = d^T U_k \Sigma_k^{-1}$$

to fold in a new document vector $d$, or

(9) $$\hat{t} = t V_k \Sigma_k^{-1}$$

to fold in a new term vector $t$.

### D.5 Implementation issues

Besides theoretical refinements, also practical implementation issues have been studied for LSI. For instance, automatic document preprocessing, tree-based methods for similarity search [6], and distributed implementation [7] have all been investigated. The existing knowledge base on implementation aspects of our project can all be brought to bear in our project.