Log Analysis aided by Latent Semantic Mapping

Submitted in partial fulfilment
of the requirements of the degree of

Master of Science

of Rhodes University

by
Stephanus Buys

Grahamstown, South Africa
April 2013

Abstract

In an age of zero-day exploits and increased on-line attacks on computing infrastructure, operational security practitioners are becoming increasingly aware of the value of the information captured in log events. Analysis of these events is critical during incident response, forensic investigations related to network breaches, hacking attacks and data leaks. Such analysis has led to the discipline of Security Event Analysis, also known as Log Analysis.

There are several challenges when dealing with events, foremost being the increased volumes at which events are often generated and stored. Furthermore, events are often captured as unstructured data, with very little consistency in the formats or contents of the events.

In this environment, security analysts and implementers of Log Management (LM) or Security Information and Event Management (SIEM) systems face the daunting task of identifying, classifying and disambiguating massive volumes of events in order for security analysis and automation to proceed.

Latent Semantic Mapping (LSM) is a proven paradigm shown to be an effective method of, among other things, enabling word clustering, document clustering, topic clustering and semantic inference.

This research is an investigation into the practical application of LSM in the discipline of Security Event Analysis, showing the value of using LSM to assist practitioners in identifying types of events, classifying events as belonging to certain sources or technologies and disambiguating different events from each other.

The culmination of this research presents adaptations to traditional natural language processing techniques that resulted in improved efficacy of LSM when dealing with Security Event Analysis.

This research provides strong evidence supporting the wider adoption and use of LSM, as well as further investigation into Security Event Analysis assisted by LSM and other natural language or computer-learning processing techniques.

Acknowledgements

First and foremost, I would to thank my lovely wife Taryn, and my wonderful sons, Cael and Egan, for their love, indulgence and tolerance during the past two years, for being such a fantastic support to an often absent, absent-minded and distracted father and husband.

I also extend sincere thanks to Dr. Barry Irwin, my supervisor, for his support and guidance throughout this process. As a first time researcher, I have found this process bewildering, frustrating and challenging at times, but also extremely gratifying and acknowledge that it would certainly not have been possible without his assistance.

A special word of thanks is also extended to the following persons: Yusuf Motara, my co-supervisor, for his insights and support, and for his willingness to challenge my thinking throughout the process; and to my colleague, Joon Radley, for his valuable insights and assistance.

I would also like to give a special word of thanks to Matthias Neeracher from Apple Inc. for his assistance and support when dealing with the LSM Application Programming Interface (API), as well as to Dr. Jerome R. Bellegarda for taking the time to answer questions for a fellow researcher.

And I offer yet another thank you to both Michael Wilde of Splunk Inc. and Michael Irwin for providing additional data for my experiments.

Lastly, I would like to thank everyone else who assisted and supported me during the process: my mom, my dad - the original scientist in my life - and everyone whom I have forgotten to mention by name. Your indulgence and support made it possible for me to complete this work. I sincerely hope that this research is a testament to the sacrifices made by all of us.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

We live in a connected world. A vast amount of commercial and private activity takes place each and every second on millions of computers spanning the globe, all connected through the Internet. In this world, consumers, vendors and criminals are separated from each other only by mere milliseconds, each extending their presence through the use of radio, copper and fibre-optic networks transmitting their signals near or at the speed of light.

Unfortunately, making sense of all of this information, this Big Data (Manyika *et al.*, 2011), is increasingly becoming more of a challenge. Thousands of transactions per second may contain both good and bad intent, valid transactions as well as intentionally malicious transactions conveying the perpetrator's attempts to commit fraud, exfiltrate sensitive or secret data or attack a target system's vulnerabilities with specially crafted exploits.

All of this activity exists as temporary electronic constructs within millions of pieces of silicon which are, in their very nature, fleeting. They are the building blocks for business, entertainment, industrial and many other processes.

Unless committed to some form of record in non-volatile storage mechanisms — be it a log file capturing transaction information produced by some middleware, or a specialised security device generating records based on actively inspected network traffic as the traffic traverses its network interfaces — it is likely difficult, if not impossible, to investigate subsequently, with sufficient context, what has taken place.

In recent years, the incident rate of attacks and malicious activity perpetrated via electronic means has increased drastically. The number of publicly disclosed incidents has reached an all-time[1] high, with few reasons to believe that we will anytime soon see a diminishing of this trend.

Cyber attacks have a global impact and span many different industries. Consider, for example, the highly regarded Verizon RISK Team's Verizon 2012 Data Breach Investigations Report (Baker *et al.*, 2012) for a very good analysis of data breaches investigated in cooperation with the Australian Federal Police, Dutch National High Tech Crime Unit, Irish Reporting and Information Security Service, Police Central e-Crime Unit and United States Secret Service.

As noted in Verizon's 2012 report, the importance of electronic records to aid investigations cannot be emphasised enough. A careful reading of the text will show that real-world incident response almost *always* involves log analysis. Incident response teams rely heavily on log analysis to provide necessary insights and enable remediation, litigation or prosecution to proceed. Verizon's report dedicates a section "Of Logs, Needles, and Haystacks-Part Deux" (Baker *et al.*, 2012, p. 54) to log analysis practice.

Over and above the need for log analysis to enable post-breach incident response, real-time monitoring of events is becoming an operational reality for numerous businesses and institutions. Within certain industries and sectors, information and the systems that host it are just too sensitive to allow for problems or attacks to go unnoticed; incident response time *must* be reduced in these cases to diminish harmful business impact. This highlights the vital importance of event generation and log collection even more, but it also points to the vital need for software and hardware to enable the automatic classification, correlation and recognition of both good and bad behaviours on the monitored systems.

In 2008, renowned information security pundit Richard Bejtlich reiterated that the first and foremost element of a "Defensible Network Architecture" is that it should be "Monitored" (Bejtlich, 2008). It is an unfortunate reality that unless records are kept or, to use the vernacular, logging and auditing are enabled, it would be close to impossible for automated systems to do real-time analysis and enable alerting and automated response. A monitored network relies on close cooperation throughout the entire system's architecture. Systems need to generate events; these events in turn must be collected and interpreted correctly, after which software and processes need to be in place to enable the desired operational insight and response.

[1] Data Loss Statistics, http://datalossdb.org/statistics

Traditionally, events have been generated and captured into log files to enable troubleshooting of production issues. This is due to the fact that it is trivial for developers to add logging to their software and reflect the inner state of the software which would normally have been opaque without tools such as debuggers. Software developers now regularly adopt logging to capture and retain security and audit-related events, events which provide invaluable information to Information Security Practitioners. Not only can a log provide a record of actions that have occurred, but it can also serve as digital evidence in a legal context. Seasoned professionals like the Verizon RISK Team have invaluable experience in log analysis and are frequently able to 'stitch together' a clear narrative of the events that took place by correlating different log sources into a common narrative, primarily by using the timestamps captured within the events.

The events captured from business critical systems and infrastructure logs are often the only means of detecting more insidious types of attacks, attacks for example that exploit business logic flaws or are designed to be as stealthy and furtive as possible. The recent discovery of long-running specially crafted attacks against very specific targets has given rise to the term 'Advanced Persistent Threat' (APT) (Binde *et al.*, 2011; Silva *et al.*, 2011; Tarala, 2011). These attacks often involve zero-day (0day) exploits (Frei *et al.*, 2006), exploits that have not been seen before in the wild and often bypass blacklist-based scanning and detection technologies such as Anti-Virus or Intrusion Detection Systems. The attacks are rarely detected while in progress, but the evidence thereof often remains captured in events that were collected throughout the infrastructure and applications.

Due to advances in hardware and software, the ability to collect and monitor events on a large scale has become practical in the recent decade. Improvements in algorithms, such as the use of Map Reduce (Lämmel, 2008), in combination with modern, powerful hardware, allow us to collect, store and search more event data than ever before; improvements also allow us to retrieve and search the events in ever decreasing time. The systems continue to evolve and we have seen the emergence of Security Information and Event Management (SIEM) systems that are being integrated as a standard part of security operations in businesses and institutions.

SIEM is a collection of software components or appliances with the purpose of facilitating enterprise-wide event collection, normalisation and correlation of all security-relevant events and as such, enables operational security processes such as alerting, reporting and response. Additionally, SIEM allows operational security teams to detect and react to incidents and breaches in real-time or as close to real-time as possible.

SIEM implementers have a massive challenge on their hands when it comes to data ac-

quisition. SIEM relies upon the normalisation of events in order to enable the systems to interpret their meaning correctly, but these events are scattered throughout the enterprise and have many different formats.

Because of the massive volume of events that can be collected within an organisation with modern software and hardware, it is impractical for a security practitioner to personally interpret these events in full. However, analysts such as the Verizon RISK Team have tools and methodologies designed specifically to deal with these challenges. SIEM systems, for example, provide plug-ins or add-ons to interpret event data and enable automation of event normalisation. SIEM systems are covered in more detail in chapter 2.

This research deals with the significant problems of event identification, classification and disambiguation. Thousands or even millions of systems generate events from hundreds or thousands of kinds of software, each with potentially unique event structures and contents. Not only are security analysts faced with the daunting task of interpreting this information at never-seen-before volumes during incident response, their available tools such as SIEM may not provide all the necessary relevant plug-ins for the variety of events that their systems could generate.

Traditionally, security analysts would be forced to fall back on scripting and specialised software, using familiar utilities such as grep[2], awk[3] and sed[4] to assist with the analysis of the data or, if available, use SIEM add-ons and other types of specialised software.

1.2 Objectives of the Research

The objective of this research is to evaluate whether or not Latent Semantic Mapping (LSM) could be another valuable utility available to event specialists. There is an undeniable need for utilities, methodologies and other technologies that can assist in the processing of large volumes of event information, not just to enable search but also to assist in the classification, identification and disambiguation of such events. The paper entitled "A Comparison of Approaches to Determine Topic Similarity of Weblogs for Privacy Protection" (Wu, 2011) contains a clear definition of LSM:

> "Latent Semantic Mapping (LSM) is a way to find 'semantic' similarities between words and documents. Two words are 'similar' if they appear with the

[2]grep, http://www.gnu.org/software/grep
[3]GNU Implementation of awk, http://www.gnu.org/software/gawk
[4]sed, the Streams Editor, http://www.gnu.org/software/sed

> same or similar other words in a set of training documents. For example, baseball and pitcher are similar, while baseball and orchestra are probably quite different. Two documents are 'similar' if they contain lots of similar words."

LSM (Bellegarda, 2008) is a generalisation of the Latent Semantic Analysis (LSA) paradigm often used in the field of Information Retrieval, discussed in detail in "Latent Semantic Mapping - Principles & Applications" section 2.3. Dr. Jerome R. Bellegarda, an Apple Distinguished Scientist in Speech & Language Technologies, recognised that the paradigm applies to fields other than Information Retrieval and coined this particular term. LSM has in fact been successfully deployed in a wide variety of use-cases ranging from anti-spam measures in the Mail application on MacOS 10 (Bellegarda, 2008, p. 33) to voice synthesis and voice recognition, and even in DNA research.

This research is an attempt to apply LSM to computer-generated events, particularly focussed on challenges such as the classification, identification and disambiguation of different events from each other. The hypothesis is that there is an underlying latent semantic structure to events. In other words, there are patterns in the words, symbols and numbers that may not be immediately obvious to humans. These patterns, such as the co-occurrence of words and symbols within certain types of events, are exposed through the application of LSM and can then be used to associate events with a shared meaning (such as all logon events, regardless of which system generated them) with each other.

This research will also attempt to determine the level of effort required for a security analyst or SIEM implementer to use out-of-the-box LSM utilities, whether or not the recall performance of the tool is sufficient, and whether or not it is deployable at a reasonable scale.

Three use-cases will be evaluated which represent the perspective of either a security analyst or a SIEM implementer. The research will replicate real-world scenarios through the use of the LSM toolset supplied with MacOS 10.8 which has a command-line interface and API. Several data sources will be evaluated to closely mimic the challenges faced by analysts and implementers.

The research also includes the results obtained from the use of custom software components developed as a part of the research process.

1.3 Assumptions and Limitations

This body of work is produced to be consumed by a person familiar with common Computer Science terms and phrases and having a firm understanding of data processing concepts. It is also assumed that the reader has a robust understanding of Information Security, the technologies involved, as well as its role in supporting operational security workflow and procedures. This is of particular relevance where the research deals with SIEM and the roles that logging and event collection play in operational security.

This text does not serve as an exhaustive treatment of LSM; however, an attempt is made to ensure a practical understanding of the technology and its applicability to the research. For detailed treatment and description of the mathematical underpinnings of the paradigm, the reader is referred to "Latent Semantic Mapping - Principles & Applications" (Bellegarda, 2008).

From mathematical and theoretical perspectives, this work could perhaps be regarded as a relatively shallow exploration of the workings of LSM. Rather, the research focussed on applying the paradigm through the combination of off-the-shelf components, reviewed in more detail in section 3.3.

This research simulates scenarios encountered by security analysts and SIEM implementers, mimicking the environment through representative test data obtained from production and lab environments. As data plays an important part of the testing, this research will evaluate the performance of LSM on representative and varied data. In real-world scenarios, as illustrated in the introduction, the data may be varied, unique or even proprietary, and confidential; with this in mind, and even though this research will exercise the concepts and use-cases vigorously, LSM may still result in unanticipated behaviour with new data sets.

It is also noted that this research does not serve as a comparative analysis of LSM against other types of classifiers such as Bayes Classifiers commonly found in junk e-mail filter deployments.

1.4 Document Conventions

The following terms are used throughout this document and warrant further definition in the context of this research:

event

An 'event' is the smallest collection of data, the smallest meaningful composition of individual units, this research deals with. The individual units may consist of numbers, punctuation or words. 'Event' is the chosen term to represent a single log entry or line in a log file. Events can be contained within a single line of a log file, such as with syslog events, or may consist of multiple lines and a hundred or more words, such as with Windows Event Log[5] events.

A vigorous attempt has been made to refer to these compositions as 'events' consistently throughout this work; there may, however, be inclusions of phrases such as 'log entry', 'log message' or 'message' within certain sections, terms with which the reader may be familiar.

event source

An 'event source' is an actual collection of events, normally found in the form of a log file but also could be API based, such as with Windows Event Log. There can be many different event sources, each emitting events. In large enterprises, it would not be unusual to have thousands of event sources. In information retrieval terms, an event source would be analogous to a corpus of documents.

sourcetype

In the context of this research, we refer to the kind of event source as a 'sourcetype', the set of types of events generated by an event source. This is a term borrowed from Splunk[6], a product with which the researcher is very familiar.

Examples of different sourcetypes are "Dovecot IMAP" and "Bind", where the "Dovecot IMAP" sourcetype would refer to the collection of different kinds of events (or eventtypes, discussed next) created by an instance of the Dovecot IMAP server[7].

It should be noted that a sourcetype often contains the events generated by a single application, but sometimes may span multiple applications such as with "Windows Security

[5]Windows Event Log, http://msdn.microsoft.com/en-us/library/windows/desktop/aa385780.aspx
[6]Splunk Inc., http://www.splunk.com
[7]Dovecot IMAP, http://www.dovecot.org

Events" collected in the Windows Security Eventlog. Sourcetypes are a logical grouping mechanism. Further illustrative examples can be seen in Table 1.1.

Table 1.1: Example Sourcetypes

Sourcetype Name	Description
Bind	All types of events generated by the BIND DNS Server, typically contains events such as logon, spooler messages and errors
Dovecot IMAP	All types of events, including logon, logon failed and other types of messages generated by the Dovecot IMAP daemon
Windows Security Events	Security related events generated by a Microsoft Windows operating system or application executed on it
iSeries Audit Journal	Audit related events generated by software executed on an IBM iSeries host

eventtype

The term 'Eventtype' is used within this research to illustrate the type of event. Whereas sourcetype is the complete set or a subset of events associated with an event source, eventtype deals with the classification of the actual event, so a sourcetype may contain many different eventtypes. Eventtypes address the action or outcome described by an event. Many different sourcetypes may generate the same eventtypes even though the structure of the events may differ depending on the sourcetype's implementation details. Examples of eventtypes can be seen in Table 1.2.

Table 1.2: Example Access Control Eventtypes

Eventtype	Description
logon	All events that relate to the outcome of an attempted logon to an operating system, application or other services
logon success	The subset of logon events that describe a successful logon event
logon failed	The subset of logon events that describe a failed logon event
account created	Events recording the successful creation of a user account in an application, directory service or on an operating system
account locked	An event that reflects that an account has been locked and may not be used again until unlocked by an administrator or some other process, such as a timer

category

As will be evident from the Literature Review, LSM constructs maps with one or more categories or semantic anchors. In this research, within the appropriate context, 'category' refers to the LSM Command Line Interface (CLI) and API treatment of LSM semantic anchors.

categorisation

In the context of this research, 'categorisation' is defined as the process whereby an event is evaluated against an LSM map and assigned to an LSM semantic anchor or category.

classification

'Classification' is the process which results in events being correctly associated with the correct sourcetype or eventtype.

disambiguation

'Disambiguation' within the context of this research refers to the ability of a security practitioner to differentiate events from each other, an integral part of the identification process and normally requiring some domain knowledge of the events involved. With some eventtypes, disambiguation could be challenging without proper counter examples. The research will attempt to use LSM to disambiguate large collections of events.

identification

'Identification', an integral part of the process that results in classification, can be defined as a method of ensuring that the subject is the entity that it claims to be (Harris, 2012). In the context of this research, identification is the result of the process whereby an event has been identified as a certain pre-determined eventtype or sourcetype. For example, an event within a new data set may be identified as a "logon" eventtype.

normalisation

'Normalisation' deals with the syntactic structure of an event (Marty, 2007). The normalisation process, an important enabler for reporting, is critical for SIEM to function correctly. Within this context, normalisation describes the process whereby fields such as a source IP address or username are extracted from an event. This may be achieved by the use of regular expressions evaluated against the event or by other programmatic means. Regular expression matching and other types of processing are also used to enrich event information with more descriptive, normalised terms, often referenced from a taxonomy.

The fields extracted as a part of the normalisation process are moved through various mathematical and statistical processes to produce metrics such as a risk score, a probability for attack or something as simple as a total count of failed logons broken down by username.

A normalised successful logon event may be tagged as 'access control', 'success', or 'action' and have the username, source and destination fields extracted.

collector software

In chapter 2 the term 'collector software' is used to describe software that collects events generated by applications or operating systems which in turn transmits the events to an LM system or SIEM. Collector software normally does not generate events, although some collector software may generate events based on activity taking place within applications. OSSEC[8], a Host Intrusion Detection System, is an example of a collector software that forwards operating security events but also generates events – such as notifications that critical files have been changed – based on changes occurring on the host operating system.

event specialists

The term 'event specialist' signifies an individual who has an interest in security events in the context of the disciplines of LM or SIEM. Two roles, security analyst and SIEM implementer, have been collapsed into this term where the distinction of the type of event consumer is unimportant.

Security Event Analysis and SIEM implementation challenges are discussed in more detail in the next chapter.

[8]OSSEC, http://www.ossec.net

1.5 Document Structure

The remainder of this work contains four chapters that are structured as follows:

- Chapter 2 presents the foundations of this research. A more thorough discussion of the background leading to this research is presented, as well as an introduction to LSM that serves to establish the fundamentals needed to interpret the experimental work and subsequent results.
- Chapter 3 presents more detail regarding the design of the experiments, the data collected and analysed, as well as development undertaken to facilitate the execution of the experiments. The chapter also contains section 3.4 in which the experiments associated with the three selected use-cases are discussed in detail. An analysis of the results from each use-case experiment can also be found at the end of each use-case section.
- Chapter 4 reflects on the results presented in chapter 3. The discussion includes a revisit of the successes achieved, the metrics selected to evaluate the results, as well as an examination of the procedures used within the experiments.
- Chapter 5 contains the conclusions derived from the research. Research goals are revisited and the work that was performed is reflected upon. Possible future research that may be derived from this work is suggested.

The document concludes with an appendix containing a list of use-cases that were not addressed within the context of this research.

Chapter 2

Literature Review

This chapter will review in greater detail the core concepts and source material upon which this research is based. It consists of three distinct parts:

Section 2.1 of the Literature Review examines core concepts – events, event sources and event collections – and the role they play in enabling the operational security processes of an organisation.

Section 2.2 of the Literature Review deals with Security Event Analysis. In this section closer attention is given to the operational security process, procedures followed by security analysts and SIEM. It will also include a review of challenges related to event normalisation, correlation and the running of functional large-scale security monitoring systems.

Section 2.3 of the Literature Review focusses briefly on the Natural Language Processing field where LSA and then Latent Semantic Mapping (LSM) made their original appearances. This is followed by a more detailed introduction to LSM.

2.1 Security Events

Events are used to convey a message to the user or administrator regarding some state that has arisen within the software about which the developer needs to make the administrator or user aware.

An event, for example, could simply convey the message that the software successfully started, or an event may be generated by the operating system serving as a notification that an application terminated abruptly.

Stated another way, an event is a type of record that is generated and possibly retained on behalf of operating systems and applications for debugging, monitoring or auditing purposes used by developers, administrators and other consumers of event data such as Windows Security Events[1] (Kent and Souppaya, 2006).

Most events have an associated timestamp. Effective analysis and response often depend on the essential context that is created by analysis based on time. There is a substantial difference between the two following observations: "We had a database failure and we had a network failure" versus "We had a network failure; then we had a database failure". Although possibly falsely correlated in the second statement, an experienced analyst would triage the multiple failures more effectively and investigate the network failures first during an investigation, as this may have been the sole reason for the database failures. The absence of the time-based correlation could result in the database team searching for problems where no permanent issues may actually exist (Kent and Souppaya, 2006). Due to this special property, log files are often classified as time-series data to differentiate them from other types of normalised and structured data found in relational database systems (Sorkin, 2009).

On modern operating systems, a great deal of attention is given to logging and auditing. The event sources are often the primary means through which administrators troubleshoot operational issues. Events provide invaluable feedback to developers regarding runtime state or failure points within the code. Recent developments have led to the creation of massively parallel data processing engines that enable real-time or near real-time analysis of event-based data to enable business analytics, application performance management and to provide operational insight to their infrastructure for companies that rely on information technology for conducting their day-to-day business (Borthakur *et al.*, 2011).

Events contained in log files or event sources such as Microsoft Windows Event Log are the principal mechanisms through which software relay audit and other security-related information back to security analysts and personnel involved in operational security. These events relay both seemingly benign state changes (a possible indicator of a breach in change management processes), and malicious activity (such as an unauthorised user

[1]Description of security events in Windows Vista and in Windows Server 2008, http://support.microsoft.com/kb/947226

attempting to alter or delete information in a database table). An entire industry has developed around this field, constructing tools and products that deal with these events and facilitate real-time or near real-time correlation and response (Marty, 2011; Rowlingson, 2004).

The importance of these events is reflected by the care that goes into their collection. As timestamps are provided, operating systems are carefully synchronised with time servers, typically using NTP (Bagwill *et al.*, 1995); events are regularly transported through secure transport to dedicated hardware and software in a practice known as 'Log Management'. Log Management systems and SIEM systems often support the capability to cryptographically sign events or blocks of events and store them securely. This, combined with the care taken during collection, enables the relevant events to be used as digital evidence and serves as evidence in a legal context (Tarala, 2011).

Events are produced from a wide variety of event sources, and the storage of events also differs according to available hardware, software and the use-case for which they are being collected. What follows is a brief investigation of popular event sources; it should, however, be noted that this list is by no means exhaustive, as events can be generated by any software and be in any format.

The simplest event stores are 'plain' log files; most experienced administrators and developers can appreciate the utility presented by simple log files consisting of just text. A wide variety of command-line tools exist to facilitate searching and manipulating the event data. Examples of these tools on Unix are cat, sed, awk and grep. Text editors such as vi[2] or emacs[3] are often used to facilitate easy navigation and search. One of the major challenges with this approach, however, is that most of the tools do not rely on indexing or other methods to provide search speed improvements and thus the user's search time is closely correlated to the size of the data set, or in other words, the tools might not scale well if the data sets become too big.

Log files are often generated directly from code, but since its creation in the 1980s, Syslog (Koivunen, 2010, p. 12) has been the de-facto logging mechanism on Unix systems. Syslog was a de-facto standard for many years but eventually turned into an Internet Engineering Task Force Request for comment (IETF RFC). The current version of the Syslog standard is RFC5424 (Syslog, 2012). Popular modern implementations of the Syslog software are Syslog-NG[4] and rsyslog[5].

[2]Vim, http://www.vim.org
[3]GNU Emacs, http://www.gnu.org/software/emacs
[4]Syslog-NG, http://www.balabit.com/network-security/syslog-ng
[5]rsyslog, http://www.rsyslog.com

It should be noted that apart from creating and managing log files on Unix or Linux host by facilitating the capture and storage of events generated by software running locally on those systems, Syslog is often deployed to facilitate the capture and storage of events generated by other systems on the network. Originally, all Syslog event traffic was in the form of Syslog protocol messages transmitted to a Syslog server accepting network traffic on UDP port 514. Modern Syslog implementations also support TCP based transport as well as Transport Layer Security (TLS). TCP plays a pivotal role in signalling to the event source that the Syslog server may not be available or not accepting Syslog traffic due to an error condition on the server. There exists no similar transport layer mechanism with UDP. TLS is critical to ensure the confidentiality and integrity of Syslog traffic. Secure Socket Layer (SSL) Certificates are also often used to authenticate event sources to the Syslog server (Kent and Souppaya, 2006).

Syslog event sources often take the form of embedded systems such as network infrastructure devices. However, these firewalls, switches or routers often have very limited internal storage and rely on a Syslog server for extended event storage capacity.

Simple Network Management Protocol (SNMP) is another legacy protocol that is often used by embedded and other systems to send events to a centralised network management console. SNMP version 3, the latest version of the protocol, is defined in IETF RFC's 3411 to 3418 (SMNP, 2012). It should be noted that SNMP versions 1 and 2c are still very popular and in widespread use. SNMP events are referred to as 'traps' as they are sent by various network and infrastructure devices to a 'trap' server where they are collected and stored.

On the Windows platform, the system standard event source is called Windows Event Log[6]. Originally there were three standard log destinations: the Application, System and Security logs. Modern versions of Windows have added additional log destinations to accommodate specialised retention policies for events such as those related to Microsoft Active Directory replication. The Windows Event Log is searched primarily through an interface provided by the operating system, but events can also be accessed through APIs provided by the operating system. Additionally, several third-party collector products have been created to facilitate the collection and storage of Windows Event Log events in Log Management systems and SIEMs. Snare[7], one such product, collects Windows Event Log information and transmits it to a Syslog server using the Syslog protocol.

Internally, Windows Event Log events are stored in a proprietary data store; when these

[6]Windows Event Log, http://msdn.microsoft.com/en-us/library/windows/desktop/aa385780.aspx
[7]Snare, http://www.intersectalliance.com

events are collected and transmitted, the formats of the events are determined by the collector software. As such, a standard Windows event may have several different formats depending on the design decisions of the developers of the collector software (Karlzén, 2008).

Events collected from within database systems also share some of the characteristics of Windows Event Logs in that the format of the event is often determined by the developer of the relevant collector software (Nair, 2008). Events are often collected from internal audit tables using a provided API; on most relational database management systems, access to such tables is provided through some version of the Structured Query Language (SQL) protocol. Several database vendors also provide software and capabilities to export such events into files. Microsoft SQL Server provides several mechanisms for event collection: internal audit tables, export through a proprietary 'trace' format or event through Windows Event Log. Each of these mechanisms could provide events for the same information with differing formats.

Mainframes and other legacy systems like the IBM iSeries provide further challenges in that they may require specialised skills to enable collection of logs. Due to their heavy use in financial institutions, the auditing and logging mechanisms on these platforms are mature, often capable of very fine-grained event generation.

In modern system architectures, there could be many different so-called software layers. At a high level these are often divided into three major sections: 1) the presentation layer; 2) the middleware or application layer; and 3) the database or storage layer. Large modern applications often have multiple software components at each of these layers, connected by complex network architectures. A typical Internet banking application, for example, may have tens or hundreds of web-servers serving as the primary user interface, and a middleware layer consisting of multiple application servers which in turn interface with identity stores, databases and even mainframes. Each piece of software in these layers, and its corresponding network fabric, could produce tens or hundreds of events per second. It is not uncommon for such an architecture to generate tens of gigabytes worth of events per day (Fry, 2011).

Some or all of these events may be stored temporarily within log files and other stores on the host operating systems, although mature deployments tend to limit such usage and rather seek to centralise event collection on a Log Management infrastructure. This centralisation allows for more flexibility when it comes to search, correlation and the archiving of the data. Syslog is a popular choice for the centralisation of the network

infrastructure events. File synchronisation and batch uploads via scripting are often also brought to bear for a file-based strategy, although file-based strategies often fail to scale when events are needed for real-time scenarios or operational monitoring. The files are mainly retained for compliance and archive purposes with some use taking place to facilitate troubleshooting and fault finding (Kent and Souppaya, 2006; Rowlingson, 2004).

Events can also be stored in traditional RDBMS, although this tends to require up-front normalisation (Josephes, 2005). The advantages of this type of approach are that data retrieval is often faster and that some indexing is also possible. This becomes very important when log volumes are high. A typical scenario may involve the extraction of key fields such as source and destination network address information, a strategy which works well for mature organisations where workflow has been well-established and key fields covering most reporting and search needs have already been identified. The strategy may be referred to as a data reduction strategy, as the original event may be discarded and only key fields get stored. A thorough understanding of the event source needs to accompany such a strategy as unidentified eventtypes may be erroneously discarded. The advantage of this strategy, though, is that with a well-designed database and mature workflow and reporting requirements, the recall performance of such a system is unrivalled.

There are also specialised event stores like Splunk and other schema-less time-series databases (Bitincka *et al.*, 2010) such as Sumo Logic[8] and OpenTSDB[9]. Within such a store, the original text representation of an event is retained and indexed according to time. Keyword-based indexing also contributes to measurable improvements in search times. There are definite advantages to this strategy, such as the retention of the original event text and also the search performance.

Other mechanisms exist, ranging from large-scale compute clusters storing events within Hadoop Distributed File System (HDFS) and using Hadoop (Borthakur *et al.*, 2011; Jacob, 2012; Lämmel, 2008) to process it, through to hybrid solutions that combine file-based, database-based or time-series database-based strategies.

There are a large number of devices, applications and operating systems that generate events which are then collected and retained using various mechanisms as previously described. The formats and content of these events depend on multiple factors including the whim of the developer, the style guidelines developed by the software's architects, the nuances of the event source API or design decisions made by developers of event collector software and the subsequent event stores.

[8]Sumo Logic, http://www.sumologic.com
[9]OpenTSDB, http://opentsdb.net

Each component may introduce changes to the format of the event, and as mentioned before, it is not uncommon for a single event to have multiple formats depending on the collector software and storage. For example, most Syslog server implementations will prepend any received event with a timestamp, date and other meta-data configured by the Syslog administrator. In scenarios where Syslog information is forwarded from one host to another, it is not uncommon to end up with an event that has multiple timestamps and dates, with different timestamp and date formats.

This situation presents a major challenge to event specialists. For effective analysis and accurate reporting, fields need to be reliably extracted from events; parsers can sometimes break when even simple changes are introduced in the syntactic composition of an event (Karlzén, 2008; Swift, 2006). An event consumer can very easily be faced with tens or hundreds of different event formats within any medium or large environment.

Efforts are underway to establish log format standards. Arcsight, for example, has a standard called Common Event Format (CEF)[10] that standardises security event formats for collection in its line of products. The standard includes a common set of fields but lacks a fully-fledged, publicly available taxonomy of words with which to describe different eventtypes. CEF is designed to be a cross vendor log format, with the format specification available upon request.

CEF has an established install base as a fundamental part of any Arcsight deployment, but CEF has had very limited uptake in the field, although to be fair, the standard is still maturing.

Another open standard that is under active development, sponsored by the MITRE Corporation, is Common Event Expression (CEE)[11]. CEE describes the format to be used for events and allows for extensibility on the part of software developers. It is accompanied by other efforts such as the CEE Dictionary and Event Taxonomy (CDET)[12], a taxonomy that provides common words and phrases to assist in categorising events into eventtypes.

Efforts such as CEE, which is designed to facilitate identification, classification and categorisation, could make a massive contribution to the field as they could drastically reduce the complexity and time needed for the normalisation and collection processes.

In the absence of standards-based events or without the availability of software to assist in the process of normalisation, an event specialist may have no choice but to manually

[10] Arcsight: Common Event Format, http://www.arcsight.com/solutions/solutions-cef
[11] Common Event Expression, http://cee.mitre.org/about
[12] CEE Dictionary and Event Taxonomy, http://cee.mitre.org/language/event.html

identify, classify, categorise and then normalise events. This will likely be an extremely daunting tasks, even with only a handful of eventtypes to deal with, because the complexity is not only affected by the formats and various types of events, it will be further compounded by additional complexities encountered with the tools available to the individual. SIEM implementers have no choice but to normalise all relevant events and assure the availability of the necessary fields to the software to enable automation. This process is somewhat eased by collectors and adaptors shipped with the products that should cover the most common event sources (Shenk, 2010).

The syntactic structure of events can also vary widely. Syslog events are mostly contained within a single line of a log file. The complete event normally needs to fit into a UDP packet, thus limiting the size and possible content that such an event could contain. It is not uncommon to encounter multi-line event formats, although these are rarely transported via or stored in Syslog. Log files from application middleware, such as events generated by Java[13], may sometimes mimic the Syslog format but typically also contain large multi-line events with stack-trace and other debugging information. Event consumers need to be cognisant of the fact that traditional Unix command-line-based strategies may fail when dealing with these log sources. In production, these events tend to be encapsulated within Extensible Markup Language (XML) or JavaScript Object Notation (JSON) type messages or stored in proprietary structures within traditional RDBMS or time-series data stores.

Event formats may also differ between various versions of software. Furthermore, software vendors may not subscribe to a common event format across their product range; it is, therefore, not uncommon for event formats to vary widely between different software products from even a single vendor.

SIEM systems and other automated alerting and reporting engines are highly susceptible to false-negative reporting when software updates change the format of the events. Report fields may no longer be captured as before, a big problem for any product that follows a data reduction strategy and does not archive or retain the original events after parsing. This susceptibility could create a desperate situation when critical updates to infrastructure or operating system software need to be delayed for testing and patching of the monitoring infrastructure to accommodate new event formats.

[13] Java Logging Technology, http://docs.oracle.com/javase/6/docs/technotes/guides/logging/index.html

```
03/11/10 01:12:01 PM
LogName=Security
SourceName=Security
EventCode=540
EventType=8
Type=Success Audit
ComputerName=SERVER2003DC02
User=SYSTEM
Sid=S-1-5-18
SidType=1
Category=2
CategoryString=Logon/Logoff
RecordNumber=699765
Message=Successful Network Logon:
   User Name: SERVER2003DC02$
   Domain: 2003DOMAIN
   Logon ID: (0x0,0x2F0B5729)
   Logon Type:3
   Logon Process: Kerberos
   Authentication Package: Kerberos
   Workstation Name:
   Logon GUID: {815e01af-999a-9d76-97c5-4f724dc0583d}
   Caller User Name: -
   Caller Domain: -
   Caller Logon ID: -
   Caller Process ID: -
   Transited Services: -
   Source Network Address: 10.0.1.202
   Source Port: 4660
```

Figure 2.1: Sample Splunk Logon Success Event

```
   PROD-MFS1.acmetech.com
      MSWinEventLog
      4     Security
      233   Fri Sep 04 18:35:32 2009
      540   Security
      ANONYMOUS LOGON
      Well Known Group
      Success Audit
      PROD-MFS1   Logon/Logoff
            Successful Network Logon:     User Name:      Domain:      Logon
ID: (0x0,0x10DB01F2)     Logon Type: 3     Logon Process: NtLmSsp
Authentication Package: NTLM     Workstation Name: FCDC01     Logon GUID: -
Caller User Name: -     Caller Domain: -     Caller Logon ID: -     Caller
Process ID: -     Transited Services: -     Source Network Address: 10.1.1.42
Source Port: 0
      204
```

Figure 2.2: Sample Snare Logon Success Event

```
   Sep 18 11:08:35 ncorpnode1 EvntSLog:
      Fri Sep 18 11:08:35 2009
      12231849    NCORPNODE1
      Success Audit
      540   Security
      Security    Logon/Logoff
      ACME\iversa Successful Network Logon:    User Name: iversa    Domain:  ACME
Logon ID:  (0x0,0xFD8FB053)    Logon Type: 3    Logon Process: NtLmSsp
Authentication Package: NTLM    Workstation Name: FINANCE108    Logon GUID: -
Caller User Name: -    Caller Domain: -    Caller Logon ID: -    Caller Process
ID: -    Transited Services: -    Source Network Address: 10.20.72.149    Source
Port: 0
```

Figure 2.3: Sample Monitorware Logon Success Event

```
Sep 18 04:02:10 acmescout1 ossec: Alert Level: 3; Rule: 5501 - Login session
opened.; Location: (acmescout3) 10.20.4.123->/var/log/auth.log;  Sep 18 04:02:09
acmescout3 su\[20178\]: pam_unix(su:session): session opened for user cactidat by
(uid=0)
```

Figure 2.4: Sample OSSEC Logon Success Event

Figures 2.1 to 2.4 show examples of different events of the same eventtype (a successful logon). It is evident from these figures that the same information may be represented in many different ways. The first three examples are for exactly the same eventtype, 540 (Microsoft, 2012), which indicates a successful logon to a Windows domain, as witnessed by the strings "540 Security" in Monitorware[14], Figure 2.3, and Snare, Figure 2.2, as well as "EventCode=540" in Splunk, Figure 2.1. It can clearly be seen, however, that the format of these events differs widely due to implementation differences on the side of the collector software. An example of a non-Windows logon success event collected using OSSEC is shown in Figure 2.4.

2.2 Security Event Analysis

Monitoring is one of the most critical aspects to operational security. In a world of zero-day threats, an active understanding of a network's characteristics and usage patterns is critical; if accomplished, unknown threats can be detected through processes that enable anomaly detection. Understanding how an IT infrastructure gets used, what is normal and expected, is absolutely critical. Monitoring enables this type of real-world operational understanding of an environment.

Monitoring can be achieved through many means:

Application performance monitoring is often critical to running large-scale service-oriented architecture systems. Events can be gathered from within hypervisors hosting virtualised environments – from deep packet inspection, application logs and many other sources – and analysed in tools such as Splunk and CA APM[15].

Infrastructure monitoring deals mainly with operating systems and networking devices to ensure uptime and uninterrupted service. Events of this nature are often collected using SNMP directly from operating systems through operating system self-reporting or collector software installed on the systems. Popular systems include Microsoft SCOM[16] and simpler tools such as MRTG[17].

[14]Monitorware, http://www.monitorware.com/en

[15]CA Application Performance Management, http://www.ca.com/us/application-performance-management.aspx

[16]System Center Operations Manager, http://www.microsoft.com/en-in/server-cloud/system-center/operations-manager.aspx

[17]MRTG - The Multi Router Traffic Grapher, http://oss.oetiker.ch/mrtg

Analytics is also possible through careful correlation of various applications and monitoring of events (often enriched by business logic events generated specifically for the purpose of conveying extra information to the analytics processes). Analytics enables businesses not just to achieve insight into their infrastructure for the purposes of troubleshooting and fault-finding or performance planning, but also to obtain real-time or near real-time insight into the current state of their business. This is a very exciting capability upon which executives and business managers rely heavily in environments where business is primarily enabled through the use of IT infrastructure, such as with online retailers.

With analytics, events from a web server, enriched with data from an enterprise resource planning (ERP)[18] system such as SAP can show an executive in real-time what his customers are purchasing or the most popular items available through the site over a specified time period.

Events from these processes form part of a larger, effective operational security program. Operational security may archive and store events generated by applications and middleware for compliance purposes, or use the events in fraud investigations. Forensics rely heavily on the events being available *and* trustworthy (Shenk, 2010).

Over and above these traditional use-cases, well-understood and accurately interpreted events give unprecedented security insight for unknown threats. A sudden spike in CPU utilisation on an otherwise idle machine may be an indicator of a breach or of a malware infection.

A shift in transaction patterns may indicate a business logic exploit in progress. Hundreds of small transactions from a wide geographic area targeting the same account, where in the past transactions were centred around a limited geographical footprint, is one example of such a pattern. This very specific type of detection is possible when events are enriched with data such as geo-ip location (Chew *et al.*, 2008; Hayden, 2010; Payne, 2006).

Data has become more valuable as the dependence on computerised commerce has increased. Personally identifiable information is very sensitive and often protected by various agreements and laws. Unless specifically monitoring for unauthorised or unanticipated use of a database hosting such information, the only hint that it might have been copied could be a sudden increase in egress network traffic from an unanticipated host. By recording network utilisation, these kinds of traffic-based anomalies can be detected.

[18]See SAP HANA, http://www.sap.com/solutions/technology/in-memory-computing-platform/hana/overview/index.epx

Monitoring of system use and audit events may bring multiple best-practice or policy violations to light. Best practice dictates that privileged accounts and service accounts be monitored; privileged accounts, however, should never be used in production. Operation security monitoring can discover events such as interactive logons using service accounts, account sharing or logon with and use of privileged accounts.

One of the most basic monitoring scenarios, and one of the few types of reports that do not require any event normalisation, is verifying the presence of events. If an event source suddenly stops emitting events, one would not need excessive detail about the content of the events to initiate troubleshooting. Certainly, a configuration change or system failure may have caused this change. However, it is ironic to note that many organisations do not have even this most basic monitoring capability due to logging and auditing not being configured or enabled in the first place (Baker *et al.*, 2012).

This leads us to the first and foremost challenge for operational security monitoring, or for that matter, any monitoring or post-incident analysis. In order to be detected, event generation needs to be enabled and events need to be collected and retained. It may seem a trivial point, but as learned through practical experience, this frequently turns out to be very challenging. First, multiple hurdles must be overcome, ranging from convincing businesses or the relevant system owners to enable event generation, to overcoming technical hurdles caused by the implementation or development of collector software (Chuvakin *et al.*, 2010).

Secondly, sensible centralisation and retention policies go hand-in-hand with event acquisition. Security analysts may need to revert to highly specialised forensic software and recovery methods in an attempt to piece together the narrative of an incident, if it is at all possible (Tarala, 2011).

Another simple type of analysis, accomplished without the need for normalisation, requires the measurement of the volume of the events. If, for example, a web server generates web access logs that are rotated and archived daily, and the normal size of such logs are around 20MB log per day, a sudden spike to 200MB or more may direct an analyst or administrator to more closely examine the content of such files. Line counts or event counts may also prove to be useful (Baker *et al.*, 2012).

With the use of scripting to handle multi-line events, or by focussing on single-line event sources, an analyst may also use search tools such as grep or awk to search for certain keywords or for particular patterns. Searching can be combined with event counts to produce reports: for example, the number of events with the words ‘failed logon’ occurring

in a log file for a single day. In order for such a report to be accurate, an analyst would need to be familiar with and confident in the content log files. Without proper normalisation, the items may quite easily be misclassified. For example, an attacker may attempt a logon with the username 'logon success'. Naive log search could very easily lead to misreporting on an occurrence like this (Benton *et al.*, 2006).

A more robust approach would be to search for certain patterns using tools that support query constructs like regular expressions (Goyvaerts and Levithan, 2009). This definitely requires some familiarity with the events within the event source and brushes up against what would be required during the normalisation process.

Where experience or familiarity with an event source is lacking, an analyst or implementer would need access to robust product documentation, although this may not be available in great detail for all products' logging and auditing mechanisms. It should also be noted that what is written does not always reflect the true composition of events. Windows Event Log is an excellent example of a well-documented standard, at least with regards to the System and Security logs, where implementation details within the collector software may still produce unforeseen problems with the normalisation process.

In most instances, however, an event specialist needs to revert to normalisation to make sense of information captured in events.

It may be the case that an analyst or implementer is dealing with an event source that is common, well-understood and stable for some period of time. For example, before Microsoft Windows 2008, when dealing with Windows NT 2000 and 2003 based systems, an analyst could rely on searching for event code 528 and 540 to monitor successful logon events on Windows systems. In such an environment, the probability that commercially off-the-shelf software exists to assist in the normalisation of these events would be relatively high. This software would be in the form of adaptors, plug-ins or add-ons for SIEM or Log Management systems or other parsing software. There was, however, a period after the release of Windows Vista and the introduction of the event with event code 4624 that such software may have produced false positives[19].

It must be underscored that there is truly no substitute for experience, training and access to complete up-to-date documentation when dealing with normalisation. In the example above, an analyst without up-to-date knowledge of Windows Event Logs could very well

[19]Description of security events in Windows Vista and in Windows Server 2008, http://support.microsoft.com/kb/947226

have missed the new event code and not accounted for successful logons from Windows Vista, Windows Server 2008 and upwards (Chuvakin, 2010).

A worthwhile and sometimes essential first step in the normalisation process is timestamp recognition. Timestamps come in multiple formats and may or may not include millisecond accuracy, time-zone information, the year that the event was generated as well as a slew of other pertinent information. Time truly brings event information to life, as can be witnessed in Figure 2.5, allowing for correlation between different event sources and enabling the narrative of an incident to be reconstructed. A timeline, for instance, might clearly illustrate the effect on an outgoing email queue when a network link went down (Stearley *et al.*, 2010).

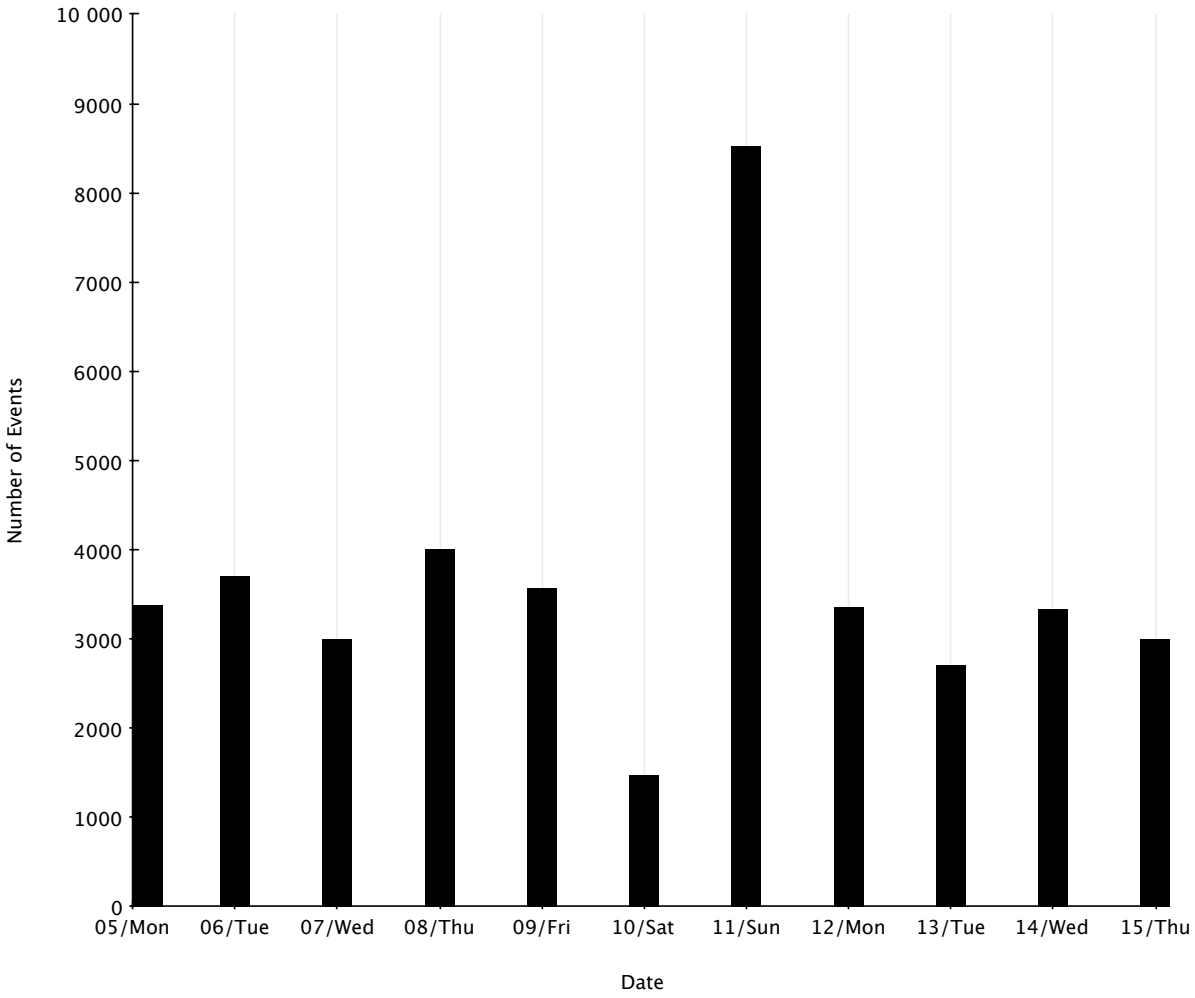

Figure 2.5: Time Series Visualisation

After successfully handling timestamp information, an analyst may need to extract certain fields from an event. Common fields in a security context would include items such as

usernames, hostnames, source or destination IP addresses. Once extracted, an analyst can create metrics such as the following: on average, users connect to the web fronted 10 times per day, while user Alice connected to the web fronted 200 times a minute for a period of 20 minutes. This anomaly would definitely be a cause for investigation: Was there a brute force attack using Alice's account? Was there some business impact during the period? Is Alice's incident a red-herring or misconfiguration? Did something else noteworthy or anomalous occur during the same period?

Field extraction may be achieved in several different ways depending on the platform and the event specialist's available tools. In a SIEM implementation scenario, the SIEM implementer is normally constrained by the framework provided by the SIEM software. While Regular Expressions (Goyvaerts and Levithan, 2009) named fields are often used, custom query languages may also be developed for use of the products. The complexity of the extraction process has a large impact on the implementation complexity of SIEM: if specialised programming skills are needed, each new eventtype for each event source may require real effort to normalise (Secmon, 2012). In a data reduction scenario, such as when loading fields into a RDBMS or into an SIEM powered by an RDBMS, it may be necessary to expend extra time and effort to ensure all possible iterations of eventtypes within an event source are covered to not accidentally discard data that may be required later.

```
Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<dxadmin>, method=PLAIN,
rip=192.168.2.10, lip=192.168.1.10, TLS

[field_extraction]
regex = (?P<timestamp>\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2})[^=]
+=\<(?P<username>[^\>]+)\>(?:[^=]+=){2}(?P<src_ip>(\d{1,3}
\.){3}\d{1,3}),\slip=(?P<dest_ip>(\d{1,3}\.){3}\d{1,3}),
\s(?P<type>\w{3}){0,1}
```

Figure 2.6: Example Field Extraction Using Regular Expressions

When dealing with log-file based events or a time-series data store where the original event text is retained, an event specialist may only need to extract a handful of fields relevant to the current metric under investigation. This greatly reduces the potential workload: as all fields do not have to be catered for, additional fields may be brought to bear at a later stage and used against the stored original events. This drastically reduces the implementation time for such systems, but the risk still exists that not all relevant events were normalised to cover for all variations of an eventtype in an event source (Splunk, 2012).

The challenge of normalisation is increased drastically at scale. A handful of events may easily be normalised as illustrated, for example, in Figure 2.6. This regular expression would result in the fields described in Table 2.1.

Table 2.1: Field Extraction Results

Field	Value
timestamp	Mar 31 23:48:20
username	dxadmin
src_ip	192.168.2.10
dest_ip	192.168.1.10
type	TLS

To extract destination address and username information from a Dovecot IMAP address, in the example extracted as the fields *username* and *dest_ip*, may require relatively little effort from an analyst. However, what if an update to the software introduces new event variation to the ecosystem? Or even more probable, what if the analyst is in a heterogeneous environment with multiple operating systems and applications, and the task at hand is to extract username and destination fields from all events of the eventtype successful logon, regardless of event source?

2.2.1 Incident Response

What follows is a brief look at some of the actions a security analyst may follow when doing incident response, provided in order to illustrate the manual process that an event specialist may need to follow and to explore how key technologies affect workflow. These examples focus on searching and interrogating available events and do not deal with ancillary processes to incident investigation such as proper handling of digital evidence. The reader is referred to (Cichonski *et al.*, 2012; Federal Communications Commission, 2002) for more detail on the larger process.

The following approaches are also product or technology agnostic and represent strategies rather than step-by-step technical guides. We assume that the analyst is experienced and knows the eventtypes and event sources that are targeted.

To start, an analyst may do ad-hoc searching of known keywords in the space. For example, when dealing with successful logon events, an analyst may search for words such

as ‘logon’, ‘login’, ‘connected’, ‘success’, ‘accepted’, ‘granted’, and ‘allowed’. There is certainly a definite risk that the sought after events may not contain these words, and also that the set of events available to the analyst may not be representative of all the variations of this eventtype.

Blacklists may also be deployed to eliminate events from search results that are true negatives. Several constructs exist in query languages to assist an analyst; for example a query such as ‘NOT failure’ may present the analyst with a result set containing a smaller set of events that need to be processed, having eliminated any events that definitely may be excluded.

Once keyword search and blacklists have reduced the number of possible events, manual inspection and iteration may be required. An analyst may manually iterate through the available events if time and set size allows; however, it is not uncommon on large production systems to still have hundreds, thousands or possibly even millions of events to inspect.

Visualisation is another incredibly powerful tool that an analyst may use. In the absence of exact knowledge of the content or structure of events, a pattern within in the data may reveal itself and assist the analyst in further minimising the possible search set, such as looking only at events within the same time range as a sudden spike in volume in the remote access server log events. For example, in Figure 2.5 the sudden spike in volume on the Sunday may indicate that an attack took place.

Regular Expressions may also be brought to bear. If, for example, an analyst recognises that most logon events have some form of “user Alice logged on successfully” or “access denied for user Alice”, a regular expression (regex) pattern may help refine the search.

In the face of very large volumes of events and the resultant need to search multiple years’ worth of application data, an analyst may have no choice but rely on *sampling* of the data, guided by experience and skill. It may not be practical or possible to inspect all candidate event data; sampling, driven by visualisation, will likely be the preferred strategy.

An analyst may also refer to other professionals, to the administrators of the system or to product documentation to obtain additional information regarding eventtypes for an event source.

Careful documentation should be created and maintained as understanding of the event source in question develops. Outliers and anomalies may reveal themselves during an analysis and will need to be accounted for.

If an environment is properly monitored and patterns associated with its systems well understood, unusual behaviour may readily be identified, tremendously assisting in reducing the search set size with which the analysts must contend.

Lastly, using tools and products to assist in the identification of well-known eventtypes allows an analyst to leverage off the domain knowledge and experience of other professionals and developers, vastly improving the potential quality of the results.

2.2.2 SIEM

Security Information and Event Management (SIEM) systems such as HP Arcsight[20], McAfee NitroSecurity[21] and RSA Envision[22] attempt to accomplish event aggregation and correlation on a large scale. A typical SIEM deployment consists of a centralised data store and correlation engine. The correlation engine analyses events in real-time or near real-time to detect security events. The central console may also support extensive alerting, reporting and monitoring functions. A SIEM deployment relies heavily on normalisation of events to extract the correct fields upon which the correlation engine depend (Shenk, 2010).

In the SANS paper "Implementing the 20 Critical Controls with Security Information and Event Management (SIEM) Systems" (Tarala, 2011), the need for SIEM to act as the central aggregator and 'brain' for event information is highlighted. SIEM is a valuable enabler of the 20 Critical Controls.

It is not uncommon to find several instances of collector software, possibly even ranging into the hundreds to thousands, to facilitate the extraction and secure transport of events from event sources to the SIEM data store and correlation engine. Event normalisation can take place either at the central SIEM through the use of add-ons or plug-ins that transform the data and store it within a database, or as run-time components that transform raw event text into normalised data during reporting or correlation processing. Events may also be transformed into the necessary normalised forms through the collector software.

Automated systems such as SIEM rely on accurate normalisation and classification of *all* target log files and events. These systems rely on a large investment of analyst, implementer and developer time to reliably classify and normalise the feeding event sources.

[20]HP Arcsight Security Intelligence, http://www.hpenterprisesecurity.com/products/hp-arcsight-security-intelligence

[21]McAfee NitroSecurity, http://www.nitrosecurity.com

[22]RSA Envision, http://www.emc.com/security/rsa-envision.htm

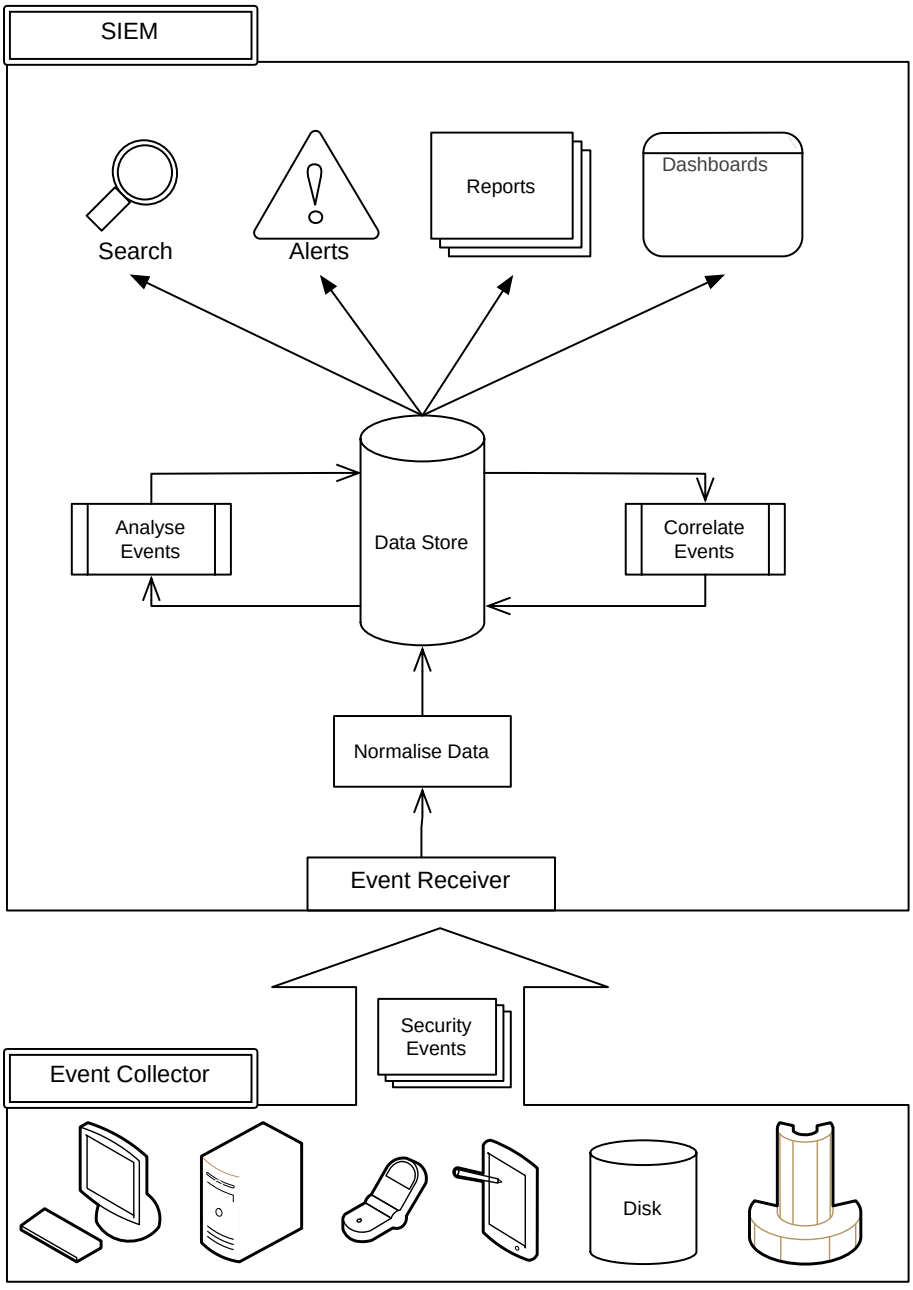

Figure 2.7: Diagram of a typical SIEM architecture

The system becomes vulnerable to change as rules and expressions that were employed to classify or normalise events might not work on new versions of software or on events within the data that were not available to developers at the time of initial normalisation process (Secmon, 2012; Splunk, 2012).

SIEM is a growing market: more and more businesses and enterprises are realising the value of visibility and the benefits of monitoring when it comes to operational security. SIEM can provide powerful insight not previously possible, especially in a large environment. Though SIEM may potentially be superseded by products that cater for more than just security reporting as more and more businesses realise the value of event data, SIEM capabilities can be integrated into larger systems that provide real-time operational intelligence based on event data from a wide variety of sources.

SIEM implementers face most of the same challenges that security analysts face when dealing with new or unknown types of data, except that there is a solid chance that the normalisation burden would be greatly increased for implementers. For effective automated correlation, alerting and reporting to take place, a SIEM needs to have all essential events normalised; this burden is lightened slightly by the availability of bundled plug-ins or add-ons. Large enterprises, however, are quite likely to have bespoke and legacy applications for which no off-the-shelf normalisation may be available (Karlzén, 2008).

As has been discussed, working with event data at scale could prove to be very challenging for event specialists. Experience, skill and tools help to reduce this burden, but the process is still arduous and lengthy. Although this research is conducted from the slightly restricted and restrained point of view of event specialists, the above challenges are not unique to those professions. In fact, challenges apply to network operations dealing with heterogeneous networking environments, operational support and software development. It is the case, though, that this research focusses on professions that are likely to encounter these challenges with a higher frequency.

Regardless of role or responsibility, the questions that the individual faces still remain: Did we identify, classify, categorise and normalise all relevant events from all the in-scope event sources? Is the reporting and alerting accurate? Do they take the complete picture into account?

An extremely thorough process may still be thwarted by outlying events not being available in sample data during development, during implementation or during analysis. Software updates on event sources or collector software may bring unforeseen changes in

formats or introduce new kinds of events. Small scale or tightly controlled and regulated environments may achieve success, but the unfortunate reality is that people are limited in their capacity to deal with these challenges at scale.

2.3 Latent Semantic Mapping

As stated previously in the introduction, this research is an investigation of the viability of LSM in assisting event specialists with many of the challenges they face when dealing with events, in particular when dealing with classification, identification and disambiguation of different events from each other.

Latent Semantic Mapping (LSM), as with Latent Semantic Analysis (LSA), operates under the assumption that there is some form of underlying latent semantic structure to a meaningful collection of data and that this structure is partially obscured by word order and word choices of the author (Bellegarda, 2008). Through the application of LSM, we can derive a vector representation of concepts and data which allow us to associate data based on this underlying structure.

Before LSM is investigated in more detail, a quick look at natural language processing and information retrieval is in order; this will be followed by a closer exploration of how the LSM paradigm operates; followed by background information assisting in the formulation of the experiments and refinement of the required approach for this research. The chapter is concluded with a discussion about similar technologies and bodies of work.

2.3.1 What is NLP

Natural Language Processing (NLP) is a well-established field in computer science. With origins that can be traced back to the 1950s, NLP deals largely with the study of computer interaction with human language. Latent Semantic Analysis, the precursor to Latent Semantic Mapping, forms part of the NLP field. Techniques and insights gleaned from NLP are highly relevant to this research.

One such technique is the process of *Stemming* which is in widespread use with information retrieval systems. Readers may already have encountered stemming in their day-to-day use of search engines such as Google as it is a process through which similar words are correlated together. For example, a search for ‘bicycle’ may also search for web

pages containing words such as 'bicycles', 'bicycles' or 'cycle'. This increases the recall performance of search engines by allowing for a degree of freedom with the construction of search queries (Uyar, 2009).

Another NLP process that is within the scope of the research is the use of stop words. Stop word processes are often used to eliminate common words such as 'a', 'the' or 'that' from a text to avoid these common words from impacting statistically on analysis. This is particularly relevant with paradigms such as LSA as common words might attract an undue statistical bias away from the semantic content that conveys the meaning of the document (Manning *et al.*, 2008).

2.3.2 What is LSM

Latent Semantic Analysis (LSA) (Landauer *et al.*, 1998), also referred to as Latent Semantic Indexing(LSI) (Deerwester *et al.*, 1990), is a well-understood technique used in Information Retrieval (IR) to improve the recall performance of large IR systems, and in particular to deal with the problem arising from users searching for or attempting to retrieve data based on conceptual content. Traditional search relies on a close match between the search phrase and the document content. LSA operates under the assumption that there is an underlying latent semantic structure to the document and that word order or word choice may obscure some of that structure (Bellegarda, 2008).

Algorithms are brought to bear on the content of the document, resulting in a parameter description of the content and documents that allow application of different retrieval techniques which may be more flexible or still be accurate enough when users are not familiar with the exact word structure or style of the documents for which they are searching.

LSA tends to deal with the underlying semantic characteristics of a composition and is not meant to deal with the syntactic nature of such a composition; it has been very successful in word clustering, document or topic clustering (Gotoh and Renals, 1997), language modelling (Bellegarda, 1998), automated call routing and automatic inference for spoken interface control (Bellegarda, 2005a, 2008).

Latent Semantic Mapping is a change of terminology from LSA to underline the use of its general properties and not just the IR applications. Jerome R. Bellegarda describes it best in his monograph *Latent Semantic Mapping - Principles & Applications* (Bellegarda,

2008). LSM is "a data-driven framework for modelling globally meaningful relationships implicit in large volumes of data". It is an extension of LSA to more than just information retrieval dealing with large corpuses of documents: the monograph deals with the definition of LSM, its relationship to LSA, and it contains examples of the use of technology in the field.

For further scrutiny of the implementation details of the LSM algorithms, the reader is referred to this monograph. The text covers the mathematical underpinnings of the paradigm in great detail and goes to great length to provide examples and case-studies of its use. This research examines the implementation and use details around the paradigm and as such will not be exploring the performance characteristics or the inner workings of the paradigm.

There are three main characteristics of LSM that are of interest for the purposes of this thesis:

1. the capability that enables discrete entities (words and documents) to be mapped onto a multi-dimensional, continuous, vector space (also referred to as an n-dimensional space or n-space);
2. the fact that this mapping is determined by global correlation patterns; and
3. that dimensionality reduction is an integral part of the process (Bellegarda, 2008).

LSM should thus allow the mapping of units within events such as words, tokens and punctuation and their related compositions - the events themselves - into a continuous vector space that captures correlation patterns between these events' reduced dimensionality form. This vector-space is referred to as the LSM 'map'; after training an LSM 'map' with the relevant or appropriate training data, several clustering and closeness-measure techniques are available to evaluate new collections against existing maps or further manipulate the maps in order to refine the data and produce associations.

It is important to note that LSM relates very superficially to the true semantic nature of the compositions as it is predominantly paradigm-based on the co-occurrences of words within compositions. As such, there is no true understanding or attempt to understand actual natural language, even though the paradigm is extensively used within Natural Language Processing.

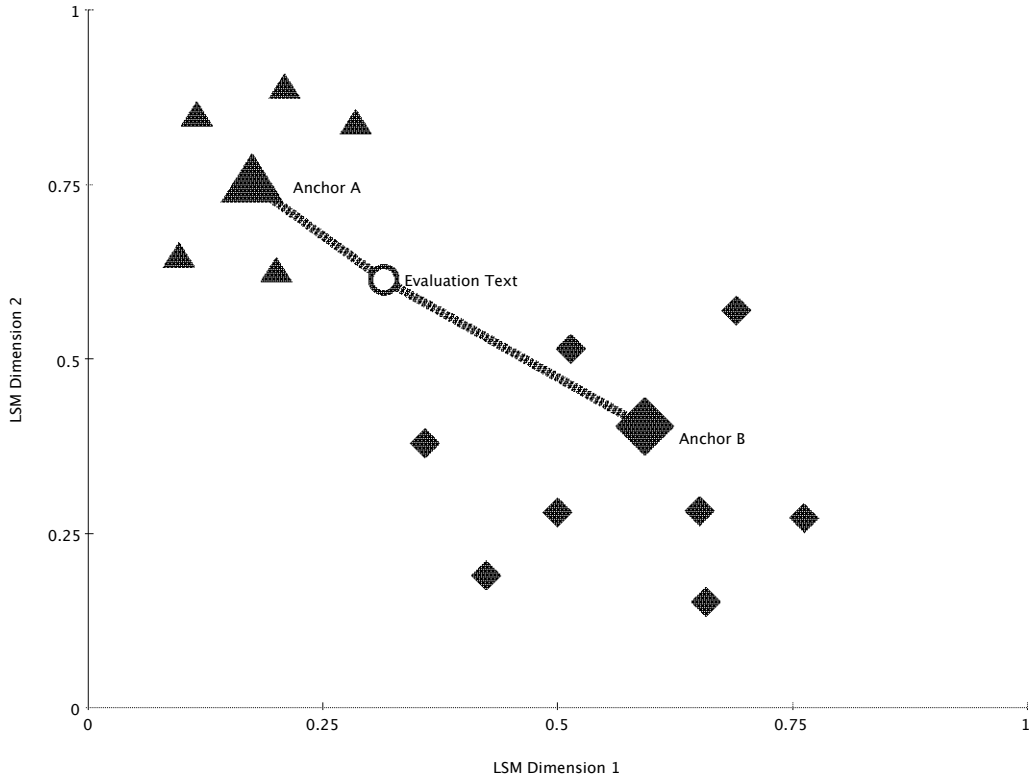

Figure 2.8: Latent Semantic Mapping

Classification

Let there be three compositions - A, B and C - as shown in Table 2.2

Table 2.2: Simple Compositions

Composition	Units
A	The cat and the hat
B	Little boy blue
C	Green eggs and ham

The LSM API or CLI is used to generate the co-occurrence matrix through a process also referred to as 'term frequency, inverse document frequency' (tf-idf) (Salton *et al.*, 1975, p. 615). The process involves counting the occurrences of words in the individual compositions versus the number of times that the words appear in the entire training set. This results in a 'bag of words' in which the word order is ignored (Bellegarda, 2005a, p. 3) (Tang *et al.*, 2003, p. 176) and creates a vector space model of the terms and compositions. The major disadvantage of relying solely on tf-idf is that the vectors resulting from the computation when dealing with large scale data sets are very sparse,

extremely large and unrelated to one another. A representation can be seen in Table 2.3.

Singular Value Decomposition (SVD) (Jolliffe, 2002) is then used to reduce the co-occurrence matrix to a set of dense singular vectors (Bellegarda, 2008, p.11) which retains most of the properties of the original co-occurrence matrix and which is a very efficient representation of the original matrix in terms of size and reduced dimensionality (Landauer and Dumais, 1997).

Table 2.3: Basic LSM Map

1	2	3	Words
2	0	0	the
1	0	0	cat
1	0	1	and
1	0	0	hat
0	1	0	little
0	1	0	boy
0	1	0	blue
0	0	1	green
0	0	1	eggs
0	0	1	ham

After SVD, the compositions are now associated with three distinct categories - 1, 2 and 3 - and the map is ready for evaluation. Another major advantage of SVD is that it enables closeness measures on the singular vectors that represent the units and compositions within the n-space. Within the LSM paradigm, as illustrated in Figure 2.8, concepts represented by these vectors that are related will measure closer to one another as compared to unrelated concepts (Bellegarda, 2008, p. 12). The correlation between closeness in LSM space and meaningful relatedness within documents has been verified extensively (Berry *et al.*, 1995)

A distance measure of composition C in Table 2.2, "Green eggs and ham", against the map represented in Table 2.3, results in the LSM distance measures represented in Table 2.4.

Table 2.4: LSM Distance Measure 1

Category	Distance
3	0.493705
1	0.259421
2	0.246874

As can be seen in Table 2.4, composition C is closest to category 3 which correlates with the training data. The distance measure of the LSM API and command-line output

results in a floating point value that represents the distance of n-space to the semantic anchor of the category. The semantic anchor is the n-space representation of the category and is sometimes used in place of LSM category. With the LSM API, the floating point values for the different categories add up to 1.0, with a larger number showing that the evaluation data is closer to the category.

Of interest in this example is the fact that composition C is not cleanly associated with category 3 although it is close enough for us to draw that conclusion. This is due to the co-occurrence of the word 'and' in both composition A and composition C.

Clustering

An LSM vector space model (n-space) can contain several dimensions and multiple semantic anchors. Anchors themselves can also be clustered together within the n-space to reduce the number of categories (Bellegarda *et al.*, 1996; Bellegarda, 2005b; Wicijowski and Ziółko, 2010). A simple illustration of such a reduction is shown in Figure 2.9

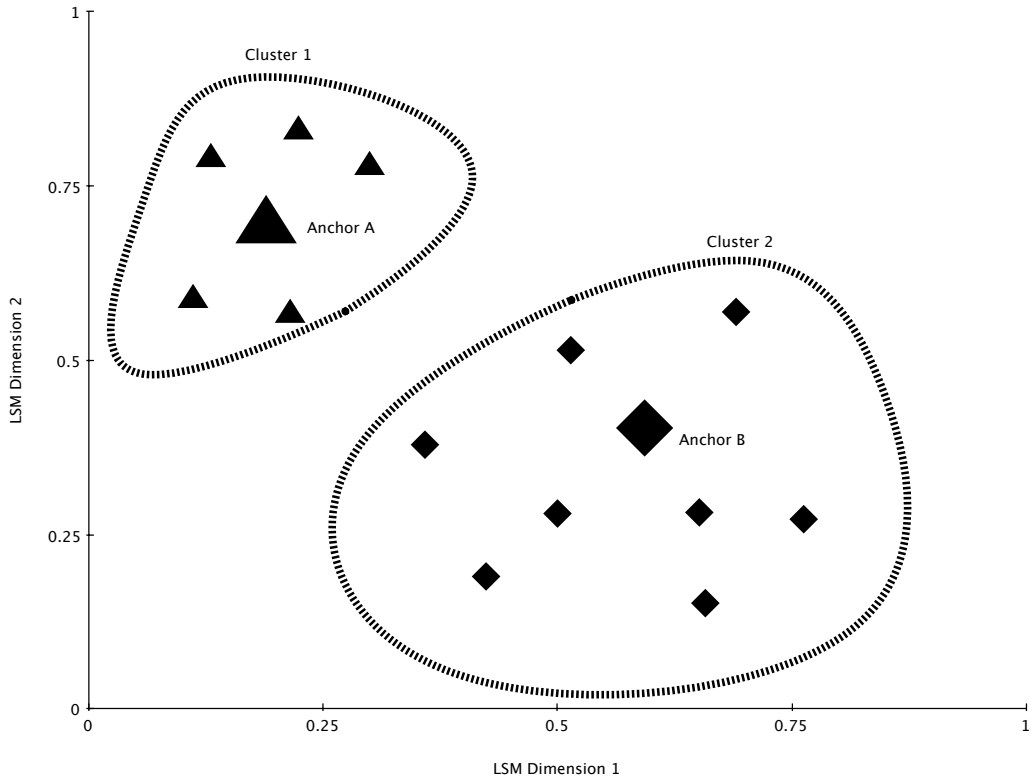

Figure 2.9: LSM Clustering

Let there be the same compositions A, B and C as shown in Table 2.2

Compositions are repeatedly added to an LSM map in random fashion, resulting in a map that contains the following matrix after 12 iterations as described in Table 2.5

Table 2.5: Larger LSM Map

1	2	3	4	5	6	7	8	9	10	11	12	Words
2	0	0	0	0	0	2	0	2	0	0	2	the
1	0	0	0	0	0	1	0	1	0	0	1	cat
1	0	1	1	0	1	1	1	1	0	0	1	and
1	0	0	0	0	0	1	0	1	0	0	1	hat
0	1	0	0	1	0	0	0	0	1	1	0	little
0	1	0	0	1	0	0	0	0	1	1	0	boy
0	1	0	0	1	0	0	0	0	1	1	0	blue
0	0	1	1	0	1	0	1	0	0	0	0	green
0	0	1	1	0	1	0	1	0	0	0	0	eggs
0	0	1	1	0	1	0	1	0	0	0	0	ham

K-means clustering is then used to reduce the number of categories, resulting in a map closely resembling the original map described in Table 2.6

Table 2.6: Clustered LSM Map

1	2	3	Words
8	0	0	the
4	0	0	cat
4	0	4	and
4	0	0	hat
0	4	0	little
0	4	0	boy
0	4	0	blue
0	0	4	green
0	0	4	eggs
0	0	4	ham

A distance measure of composition C against this map results in Table 2.7

Table 2.7: LSM Distance Measure 2

Category	Distance
3	0.493705
1	0.259421
2	0.246874

Of interest in this example is how LSM was used to reduce the number of categories in the map; this capability allows for identification of similar compositions without any prior knowledge of the content of such compositions.

For this research, compositions will be events and units will be comprised of words, tokens and punctuation that comprise those events. The capabilities in this chapter, such as LSM distance measurements and clustering, will be used in the use-cases that follow in chapter 3.

LSM Examples

LSM has seen many tests and uses applied to a wide variety of different domains; however, due to an acute absence of previous study pertaining to information security, this research sought to explore this particular vital area. LSA has had its roots in information retrieval, with extensive research having been conducted regarding its use in the IR field (Deerwester *et al.*, 1990). In the domain of information retrieval, the units are words and compositions are documents.

With its use in text summarisation (Gong and Liu, 2001), the units that are dealt with are again words, but this time the compositions are sentences. The LSM evaluation attempts to suggest a most relevant sentence, or group of sentences, to best represent a text.

In Bellegarda (2005b) and Bellegarda (2008), various use-cases for LSM have been brought to light. In the use-case of junk email filtering, the units are words and symbols and the compositions are emails. LSM is shown to perform on par with alternative solutions used within the same domain such as Bayesian classification and other strategies. In the junk email filtering use-case, two LSM categories are used: ‘spam’ and ‘not spam’. LSM has been successfully applied as a part of the anti-spam measures within Mac OS X Mail.app for almost a decade. Additionally, LSM is used in conjunction with other technologies and is not relied upon solely for classification, although it certainly plays a valuable role.

The same texts describe the use-cases of speech recognition where the units are letter n-tuples and the compositions are words, as well as speech synthesis where the units are pitch periods and the compositions are time-slices.

Within a paradigm named LaSSI, or Latent Semantic Structure Indexing, the concepts are used to rank molecules according to similarity. The units for this use-case are descriptors and the compositions are molecules, leading to the construction of a descriptor-molecule matrix (Hull *et al.*, 2001).

In Kazakow (2008), LSM is used on structured data to do ontology matching; the concepts are applied to structure data within databases, using values within object for units and the objects in the ontology as the compositions.

These use-cases illustrate LSM use in domains that are far removed from the original IR environment for which LSA was developed, underscoring the amending of terminology from LSA to LSM.

LSA plays a role in word sense disambiguation (Van de Cruys and Apidianaki, 2011) where a discussion follows concerning how to relate words and topics within a corpus of documents to derive at a 'word sense' for terms. This approach assists with disambiguation of terms to relevant topics (i.e. 'chip' in computer science versus 'chip' in baking/chocolate chip).

Although beyond the scope of this research, Tang *et al.* (2003) discusses the ability to scale Latent Semantic Indexing (LSI) to multiple nodes to enable powerful peer-to-peer (P2P) searching capabilities. The work is potentially applicable to future research where LSM scaling characteristics within large-scale log management or SIEM systems are investigated. One of the critical challenges with both log management and SIEM is that they often have to deal with massive volumes. The paper has good information related to system scalability, divisibility of the search problem, alternative strategies for information retrieval and information on updating the indexes, a major challenge as it impacts continual training of the LSM maps.

Finally, in Lobo and de Matos (2010), the researchers use LSM to organise and tag a corpus of fairy tale stories. The approach is language independent and automatically defines the number of clusters (mapped to topics and tags) within the set of documents. Within that research, the units are again words and the compositions are documents. This serves as a good introduction to the use of clustering and LSM.

Caveats

As a part of the Literature Review, some of the material highlighted potential issues when dealing with LSM, in particular, that LSM could be sensitive to composition style and polysemy (Bellegarda, 2005b), with polysemy meaning that units may convey multiple meanings. In the event of natural language, polysemy means that compositions that are closely related may be hard to classify as *not* within the same category. This is witnessed in Use-Case 1 section 3.5 and can be partially managed through the use of pre-processing (Bellegarda, 2008; Kazakow, 2008).

The 'bag of units' approach also results in retrieval issues where word order or unit order are important. This is prevalent in domains such as user interface control using spoken language (Bellegarda and Silverman, 2003) and is discussed in (Bellegarda, 2008, p. 41). The introduction of bi-grams and tri-grams assist in capturing more local context in the creation of LSM maps. Bi-grams are used throughout this research and are discussed in more detail in section 3.5.

For discussion in this research, SVD is also an offline-only process. This means that continuous updating of the vector space is not possible at present; this, however, does not affect the use-cases in the research and therefore was not explored further.

For this research, a great deal of attention was given to pre-processing as a part of the work and is discussed in more detail in subsection 3.3.4. Pre-processing is used as a measure to offset the effects of polysemy and the impact that style mismatch can have between training and evaluation data (Bellegarda, 2008). Careful training data selection and preparation plus pre-processing is used widely with LSA and LSM (Bellegarda *et al.*, 2003; Deerwester *et al.*, 1990; Kazakow, 2008; Landauer *et al.*, 1998).

Common pre-processing strategies include the use of stop words (Lobo and de Matos, 2010; Manning *et al.*, 2008; Wu, 2011), stemming (Deerwester *et al.*, 1990; Uyar, 2009; Wu, 2011) and lexicalisation. Stop words and stemming were used in this research. Lexicalisation, a process whereby names are treated as single words ('Stephan Buys' becomes 'StephanBuys', 'Rhodes University' becomes 'RhodesUniversity') (Wu, 2011), was not used in this research, as it presupposes a familiarity with the domain of the training data. Wherever possible, tokenisation was used as a substitute to generalise text and number sequences such as IP addresses.

It must be kept in mind that LSM is data driven (Bellegarda, 2005b, p. 79). The impact of the training data, its style and phenomena such as polysemy could have a substantial

impact on the quality of results and were thus continually brought into consideration for this research. Training data selection needs to be approached carefully (Bellegarda, 2005a) and as far as possible, be representative of the full breadth of the domain, as balanced as possible in each category and of sufficient volume.

LSM and LSA are almost always implemented as a component of a larger strategy (Bellegarda, 2008; Deerwester *et al.*, 1990). This work follows those recommendations as can be seen in chapter 3.

2.4 Summary

This chapter introduced Security Event Analysis and Latent Semantic Mapping in more detail. A discussion regarding the importance of Security Event Analysis commenced in thechapter 1. Within this chapter, the narrative presented a more detailed discussion of the fundamental building block of this research: the event. The discussion then examined the role that security events play in enabling operational security practice and event analysis, concluding in their role and use when operational security practice is automated in the form of technologies such as SIEM. Finally, a high-level introduction to Latent Semantic Mapping, its fundamental features and concepts, as well as established successes in practice, was then provided.

Chapter 3

Experimental Design and Execution

Chapter 3 deals with the details related to the design of the experiments as well their execution and an analysis of their results.

The chapter starts with section 3.1, an introduction to the approach that was followed during experimentation, as well as the limitations that applied in the context of this research.

Section 3.2 relates the procedures followed to obtain the data needed for the experiments, the challenges that were faced during data collection as well as details of the eventual data selected for use within the three use-cases that were examined.

Section 3.3 contains details related to the initial exploration of the paradigm which included the use of off-the-shelf components but eventually lead to the development of more sophisticated software. During this process, important insights were also achieved with regards to the application of the LSM paradigm to events and this is given additional attention in the section.

Sections 3.5 to 3.7 contains the experiments for each of the addressed use-cases. Each use-case section discusses the design of the experiments, the details of the execution of the experiments, the findings that were anticipated from the experiments as well as the results that were obtained from the experiments. Each use-case section ends with a brief discussion of its experimental results.

3.1 Approach

For this research, three experiments were conducted in order to test whether or not LSM could provide a valuable contribution to the efforts of event specialists as a part of the suite of tools that are available in the areas of log management and SIEM, as well as for incident response or forensic analysis.

Three use-cases, each representative of activities faced with challenges for event specialists, were investigated and several experiments were conducted to test the utility of LSM within these use-cases.

The experiments and their variations were not meant to be an exhaustive analysis of the use of LSM but rather an exploration of the technical implementation details. These experiments were meant to be an exploration of the possible use-cases for the use of LSM within the realm of log management and SIEM; specifically, how LSM could be applied by an analyst or operator working with the before-mentioned systems. The research, therefore, did not deal with other aspects of LSM such as the performance characteristics of the paradigm.

The experiments aimed to exercise LSM with sufficient training and evaluation data, but these were scaled down as compared to vast large enterprise log management practices. The software developed for this research was created specifically to facilitate the experiments, not designed to be robust and scalable for integration into large log management practices.

LSM will be evaluated for its usability when facing the challenges of identification, classification and disambiguation, challenges covered in the use-cases found in section 3.5 to section 3.7.

3.2 Data Description

This research considers the challenges faced by event specialists when having to deal with challenges such as classification, identification and disambiguation of events. In section 2.1 there is a detailed discussion of the need for this event data and the role that it plays in security operations. The research attempted to determine if LSM was a practical and useful tool when dealing with log management and SIEM implementation challenges

as well as the challenges faced during forensic investigations and incident response. To conduct the research and obtain verifiable results, sufficient event data needed to be collected.

Data collection for this research proved to be surprisingly challenging: as opposed to finding packet data[1][2], logs are often treated as sensitive information. From cases such as in 2006 (Shen, 2012) when the company AOL Inc. came under wide criticism and legal scrutiny for publicly releasing a large collection of search log data, public disclosure of log data has become something that large entities shun and avoid (Hafner, 2006). Over and above the relevant scarcity of publicly available log sample data, the research depended on obtaining a large variety of log eventtypes. It would not have been sufficient to attempt the research against a small collection of source and eventtypes as the consequent variety in log structure and format would have limited the real-world applicability of the research.

Several institutions and individuals within the information security industry were contacted in an effort to obtain logs, but a concern about the potentially sensitive nature of information within the events led to several abrupt dead ends. In the end, some log samples were secured from other sample sources, logs from a public network, and some logs from systems at the university. More details regarding the logs follow later in this section.

For the purposes of the research, event data from multiple sources were required. Quantity did not have a large impact on the experiments although sufficient quantities were needed to allow for proper interpretation of results. This research falls within the domain of information security research and as such, an attempt was made to collect security relevant events and log files, and an attempt was made to obtain as many access control related events as possible for the purposes of conducting the experiments related to Use-Case 1, found in section 3.5.

Splunk was used as the event store for all event samples and log files collected for the research. The total collection size within Splunk topped off at around 40,000,000 events. It is an engine for IT data, which in itself does not generate any events but created a time-series index of events that made searching and retrieval easy and facilitated the export of events into the formats needed for the experiments. Splunk has great support for multi-line events such as Windows Event Log events and alleviated the need for special management of multi-line files.

[1] OpenPacket.org, https://www.openpacket.org
[2] The Cooperative Association for Internet Data Analysis, http://caida.org

Splunk supports four types of file exports: raw file, comma separated value (CSV), JSON, and XML. Raw file support would have required special handling of multi-line events within the software developed for this research. Multi-line event encapsulation within CSV is also extremely error prone, particularly if the events themselves also contain commas.

JSON was selected as the preferred export format for training data as it is relatively simple to parse; multiple C++ parser libraries exist and JSON serialises multi-line events without the need for additional processing or handling. The software developed for the research, named LSMPP (discussed in section 3.3), supported either JSON files, single event per file (mostly prepared manually through copy-and-paste), or traditional single event per line log files.

3.2.1 Data Collection

A collection of logs from Interop Las Vegas 2012[3] was provided by Mr. Michael Wilde of Splunk Inc. Interop is a large multi-vendor conference featuring numerous technology vendors plus many prospective customers and technology decision-makers. Interop, held at least once a year for more than 20 years now, takes place in a variety of venues. One of its main attractions is InteropNet, a temporary carrier-grade network created for the duration of the conference to allow vendors to demonstrate their products in a live environment and prove inter-operability with complementing technologies.

Splunk Inc. has been providing logging and search capability to InteropNet since 2005. The data in the research dataset was captured during the period 4 May 2012 to 10 May 2012.

The data set – 262,262 events captured using the Syslog protocol – consists of single-line events from multiple technologies. The exact details of the events, their generating technologies or their structure are not known to the researcher, which makes it ideal for use in use-cases that simulate the situation often faced by a post-incident or forensic log analyst.

Bind DNS, Postfix MTA as well as Dovecot IMAP events were provided by Dr. Barry Irwin, the researcher's supervisor. Active Directory and Zenworks logs were provided by Michael Irwin, an employee of Rhodes University.

[3]Interop Las Vegas, http://www.interop.com/lasvegas

A wide variety of accessible log sample data was used to create data and examples for the research which deals with log management and SIEM; the sample events included sources such as F5, Cisco, Fortinet and many others.

Events were also generated on the researcher's own infrastructure including events from a host of services and software that can be found in the MacOS environment; as far as possible, though, the events were limited to security and system related events.

3.2.2 Data Selection

The sample events were used to generate data sets for training of the LSM classifier as well as evaluation using the LSM classifier. Where needed, special events were sought out for training purposes or a random, reduced set of events were used where certain behaviours of LSM needed testing. A discussion on these generated data sets follows in this section.

The research depended more on the variety of event compositions than on the quantity of the events; thus, it was deemed sufficient when the larger collection of events was reduced to allow for practical experimentation and evaluation to proceed.

More details related to the training data and evaluation data are provided within the Use-Case Detail section in section 3.4.

3.3 Architecture and Processing

This research focusses on the practical use and potential value of LSM to security practitioners such as event specialists. To enable the use-case experiments, an LSM processing framework had to be established.

All evaluation and development was completed on the researcher's Macbook Pro personal computer running MacOS 10.7, later upgraded to MacOS 10.8. There was no choice of platform as the LSM API and command-line interface (CLI) is implemented exclusively on the MacOS platform. This research did not attempt to delve into the algorithmic nature of the paradigm so alternative platforms and implementations were not investigated in depth. The subsection that follows presents a closer look at the early experiments conducted as a part of the research and the subsequent evolution of the processing model.

3.3.1 Early Iterations

Initial experimentation was completed with the LSM CLI[4]. The LSM CLI allows for the creation, clustering and evaluation of LSM maps, enabling researchers and developers to become familiar with the paradigm, to enjoy full access to the range of LSM capabilities, and to be provided with a reference platform when custom API- based code is developed. The LSM CLI was used extensively at the start of the research but proved to be challenging later when integration, pre-processing and analysis became important. However, later in the research cycle, the LSM CLI proved invaluable in understanding and testing LSM API code implementations and their results.

Early in the research, the value of being able to seamlessly load and pre-process training and evaluation data was realised. It also became very apparent that an effective means to link results back to the originating training and evaluation events would be necessary, particularly with LSM maps created with pre-processed events. Early attempts were made to integrate LSM CLI with Splunk, but the approach suffered from multiple issues at the integration points. One major problem, for example, was the management and linking of output data and temporary data. Each experiment run would result in more output files and temporary data to manage, as well as increased data used within Splunk. Splunk proved to be valuable when analysing initial results, but the approach was ultimately abandoned.

Next, an investigation was made into possible alternatives to the MacOS LSM API and CLI, with the evaluation of interesting projects such as Pattern[5] and other Python-based frameworks. These alternatives almost exclusively focussed on LSA, not LSM, but as discussed in the Literature Review in subsection 2.3.2 there are no fundamental differences between LSM and LSA; LSM mostly just extends the scope of possible uses of the LSA paradigm. None of the investigated frameworks, however, offered the same performance or completeness in functionality as the MacOS LSM API. This unfortunately implies that, at least for now, this research can primarily be conducted *only* on a Macintosh platform. However, there appears to be no impediment apart from time and effort – and its associated costs – that would prevent the development of fast, comprehensive, alternative implementations on other operating system platforms.

Development was started to enable use of an Objective-C based LSM classifier using the LSM API. The bulk of the understanding of the problem domain was achieved using this

[4]lsm(1) Mac OS X Manual Page, http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man1/lsm.1.html

[5]Pattern, http://www.clips.ua.ac.be/pages/pattern

software. A memory resident approach to data management was adopted to allow for easier management of results as well as to assure greater repeatability of experiment runs. Due to the exploratory nature of the work, the source code became quite unmanageable as its size increased. Ultimately, similarly to how the early CLI and Splunk integration attempts led to insight into the challenges of output and input management, the Objective-C code brought insight to another major challenge with the evaluation of the results: it became very apparent that more care would be needed in the design of the data structures to better enable linking and analysis, in particular, the ability to link evaluation strings, which are pre-processed, back to their original raw strings, as well as the ability to efficiently and consistently analyse the events after processing (for example the ability to sort, filter and display the relevant information to the user). It needs to be noted that in most use-cases the evaluation event count exceeded 1000 and that manual or iterative data analysis became challenging at that scale.

The final experiments were completed with the culmination of the software development process; this final version of the LSM classifier and surrounding processing architecture was named LSMPP. It took into account the lessons learnt from the CLI, Python plus Splunk and Objective-C prototypes.

The project was developed using C++11 and Core Foundation API (C language). The primary goal of the project was to allow for greater detail in analysis and correlation capabilities while maintaining native speeds. One of the challenges encountered during the experimentation was the correlation of the end-product of the training data and results with the original training data in its raw form. The LSMPP project allowed us to establish memory-based bindings between the training data, its pre-processed end-products and the subsequent results from the LSM Core Foundation API.

The LSM Core Foundation C API (LSM API) is developed and maintained by Apple Inc. A more in-depth treatment of the LSM API can be found in the discussion of Use-Case 1, section 3.5.

The software allows fine-grained control over the corpus of training data and how it is processed with the LSM API. A detailed discussion of the software architecture follows in subsection 3.3.2. The performance of the software also compares favourably with the Objective-C implementation and has a massive speed advantage over the Python plus Splunk as well as the CLI based approach.

3.3.2 Processing Model

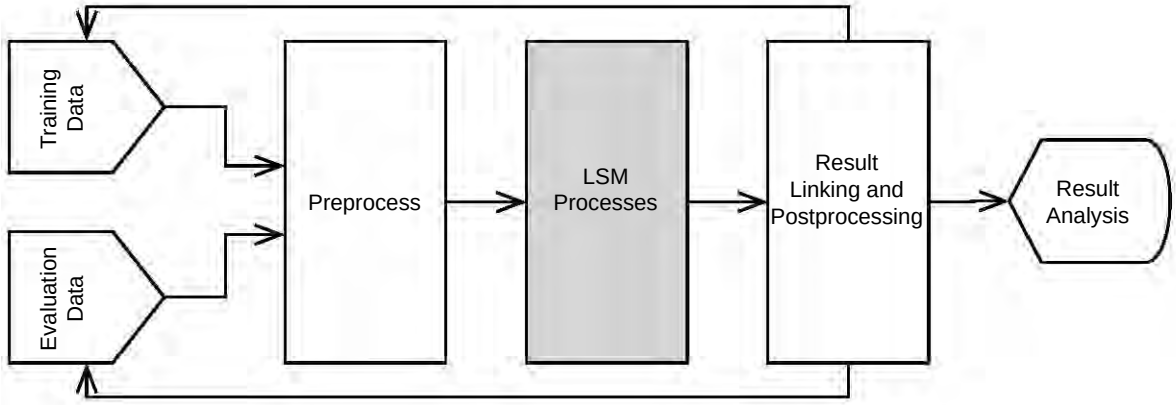

Figure 3.1: Standard Processing Model

The LSMPP software follows the processing architecture illustrated in Figure 3.1. The first aspect of the architecture dealt with the handling of the training and evaluation data.

3.3.3 Training and Evaluation Corpuses

In this architecture, all training and evaluation data is kept memory resident to facilitate linking and analysis. The data model is depicted in Figure 3.2 and highlights the most important capabilities: the ability to store the original, raw, event strings (*raw_ data*) as well as the post pre-processing variations of the events prepared for training (*training_ data*), its associated LSM category (*lsm_ category*) and event variations for evaluation (*evaluation_ data*).

All event data was stored within a *Collection* class which could be traced back to the original source file (*Source*) and the sourcetype of data in that collection (*type*), for example "Windows Security Log" or "Dovecot IMAP". All collections were associated with a *Corpus* which provided a mechanism for the software to retrieve all *Collection* objects.

Due to the memory resident nature of the architecture, the size of the training and evaluation data sets is somewhat limited. This limitation, though, did not detract from the utility of the experiments as the tests relied more on quality and type of training data as opposed to the quantity thereof.

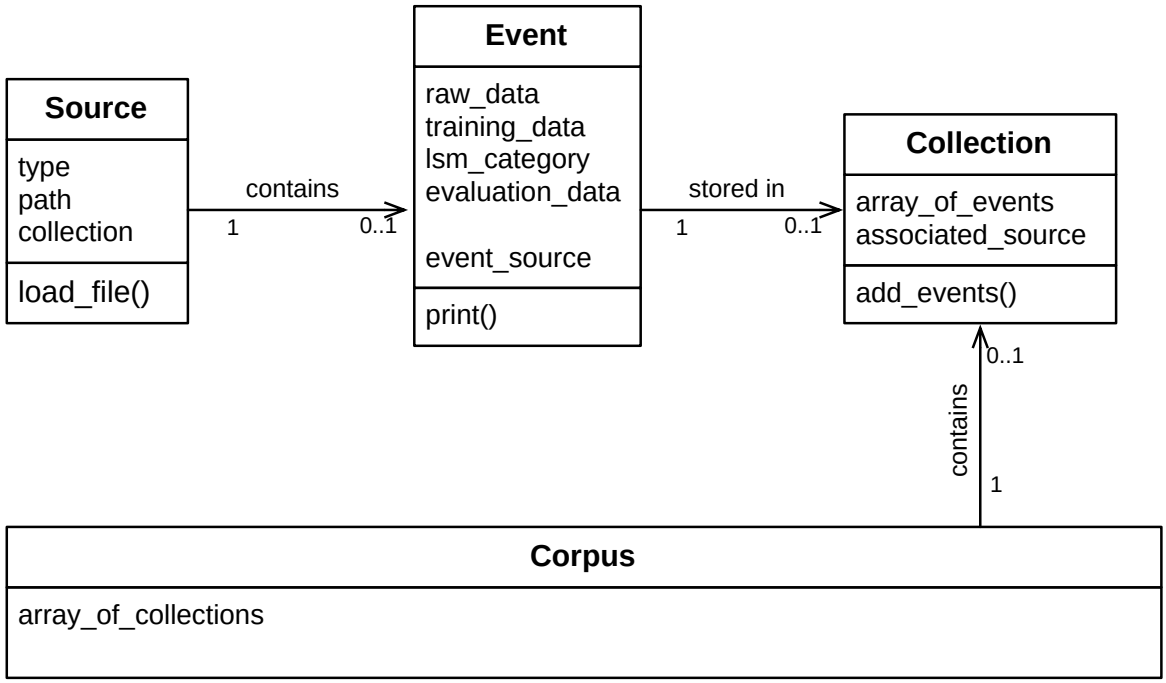

Figure 3.2: Event Data Class Diagrams

3.3.4 Pre-processing

When a pre-processor is defined, all events are processed by a set of user-defined regular expression substitutions. These regular expression substitutions can be combined in multiple fashions, but ultimately a strategy called the "AdvancedRegexStrategy" was used which combined all of the regular expression substitutions into a single processing run, a description of which can be found in subsection 3.5.2.

Lessons learnt in constructing the pre-processor became a valuable aspect of the research. With LSM, the goal is to operate on the underlying latent semantic structure of a composition, an 'event'. In the case of this research it turned out, however, that strategies suitable for other natural language processing may not be applicable to compositions such as events.

The use of stop words commonplace in natural language processing and other applications of LSA and LSM can be seen in Iwata *et al.* (2008), Lobo and de Matos (2010) and Puffinware (2010). Common stop words include articles and prepositions such as 'the', 'a', 'an', 'about', 'with', 'from' and many other regularly occurring words. Within natural language processing makes intuitive sense that the word 'the' creates a superfluous link between the two sentences: "*the* ball is *the* colour brown" and "*the* boy sat on *the* river

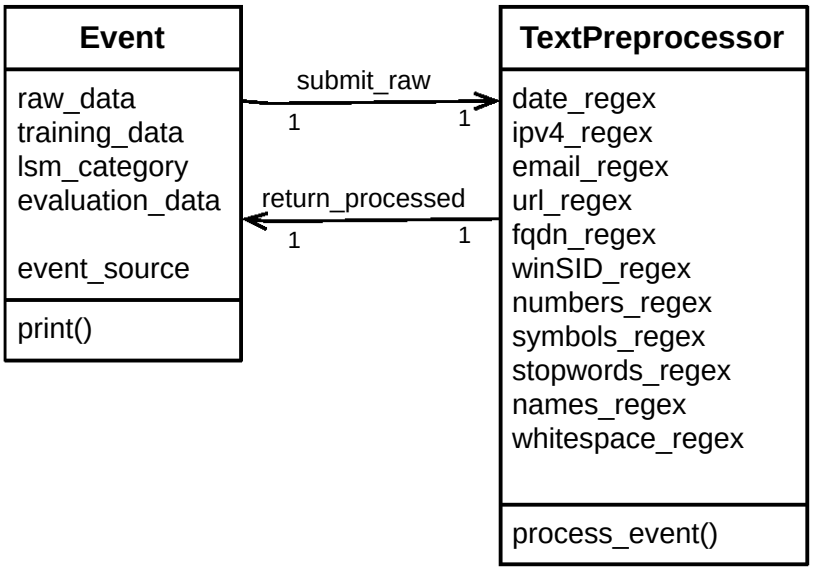

Figure 3.3: Event Pre-processor

bank". Naive implementations of LSA and LSM would associate these sentence based on the co-occurrence of the word 'the' even though the sentences do not share any meaning. With short sentences, the word frequency of 'the' in the total training set of this example is relatively high.

The first versions of the pre-processor diligently stripped out articles, prepositions and other regularly occurring words and patterns. However, this had little effect on the training and evaluation data as these particularly identified words occur relatively infrequently, and thus results were poor.

More aggressive stopping was applied in an attempt to increase the quality of the results – eliminating *all*number and punctuation as well as stop words. It was expected that the English words and strings within an event would deliver sufficient latent semantic meaning, but it had incredibly bad accuracy. Closer investigation revealed matches between IMAP logon events illustrated in Figure 3.4 and Bind events illustrated in Figure 3.5. The events were matching based on the co-occurrence of the word 'imap'. In both cases, much of the structure and meaning in the events were lost.

Stemming is a technique often deployed in information retrieval to convert words to their root words, including the dropping of suffixes and using lookup tables. In the software it would mean the conversion of words such as 'failed' and 'failing' to the word 'fail' and using only the word 'fail' during pre-processing. Stemming was a natural fit for the experiments.

```
Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<dxadmin>, method=PLAIN,
rip=192.168.2.10, lip=192.168.1.10, TLS
->
mx dovecot imap login Login use dxadmin method PLAIN
rip lip TLS
```

Figure 3.4: Dovecot IMAP Logon Event : Raw -> Processed

```
12-Aug-2010 03:59:49.170 queries: info: client 192.168.0.1#48433: query:
imap.server.example IN A -
->
queries info client query imap server example IN A
```

Figure 3.5: Bind DNS Query Event : Raw -> Processed

To improve the quality of the results, an attempt was made to identify other regularly occurring keywords and names; however, this process was abandoned due to the sheer volume of events and the large variation of words and names. It was also deemed very impractical for an event specialist as it necessitated extensive searching and data analysis even *before* LSM could be brought to bear. Lexicalisation, as discussed in section 2.3 also depends on the identification of frequently occurring terms and was abandoned in favour of the analogous process of tokenisation through regular expression substitution that the pre-processor used.

As the goal of the research is to identify whether or not there is a practical use for LSM for security practitioners, extensive manual pre-processing was deemed to be a dead end.

Aggressive use of stop words plus number and symbol elimination also appeared to shorten the events and lose an unreasonable amount of the underlying semantic meaning, as can be seen in Figure 3.5.

3.3.5 Events are not normal documents

LSM research was interrupted at this stage and a closer look at the training data followed. A closer examination of the problem the research determined to solve and a rethink of the meaning of compositions in terms of the research led to the most valuable pre-processing insight in the research.

It was realised that treating events as traditional documents was erroneous. Events, as discussed in section 2.1, are machine data. Although it is largely unstructured, it is not

natural language; event formats are limited and eventtypes within a sourcetype often share structural elements such as punctuation with one another. The LSMPP software specifically makes provision in its pre-processor to retain and tokenise as much of the latent semantic structure of events as possible. Common patterns such as IP addresses and FQDNs were tokenised instead of discarded and the end result was a closer representation of the latent structure of the message, which in turn should have been better processed and used by the LSM classifier. It can also be noted that all stop word lists were abandoned but that stemming strategies were retained.

```
Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<dxadmin>, method=PLAIN,
rip=192.168.2.10, lip=192.168.1.10, TLS
->
dateToken mx dovecot COL suspectNameToken COL logon COL user EQ LT sbuys GT
CM method EQ PLAIN CM rip EQ ip4Token CM lip EQ ip4Token CM TLS
```

Figure 3.6: Final Processing Strategy

An example of the results of this work can be seen in Figure 3.3. This "AdvancedRegexStrategy" was used for all use-case evaluations. A closer look at the post-processed event shows that punctuation and timestamps now form a part of the latent semantic structure of the event. It was reasoned that the sequence of words, as well as the unique composition of words and punctuation tokens, would provide a sufficient 'fingerprint' for LSM to work more effectively.

3.3.6 LSM Processes

Figure 3.7 illustrates the structure of the LSM classifier, interaction with events and the results. A detailed explanation of LSM API internals and their interaction within the classifier is provided during the treatment of the first use-case in subsection 3.5.3.

The LSMPP LSM classifier seamlessly supports dimensionality settings, training and evaluation mode switching, automatically linking the current training category back to training events as well as creating a link between LSM results and the evaluated event.

Once an LSM map has been created, the classifier can also handle clustering when requested by the user.

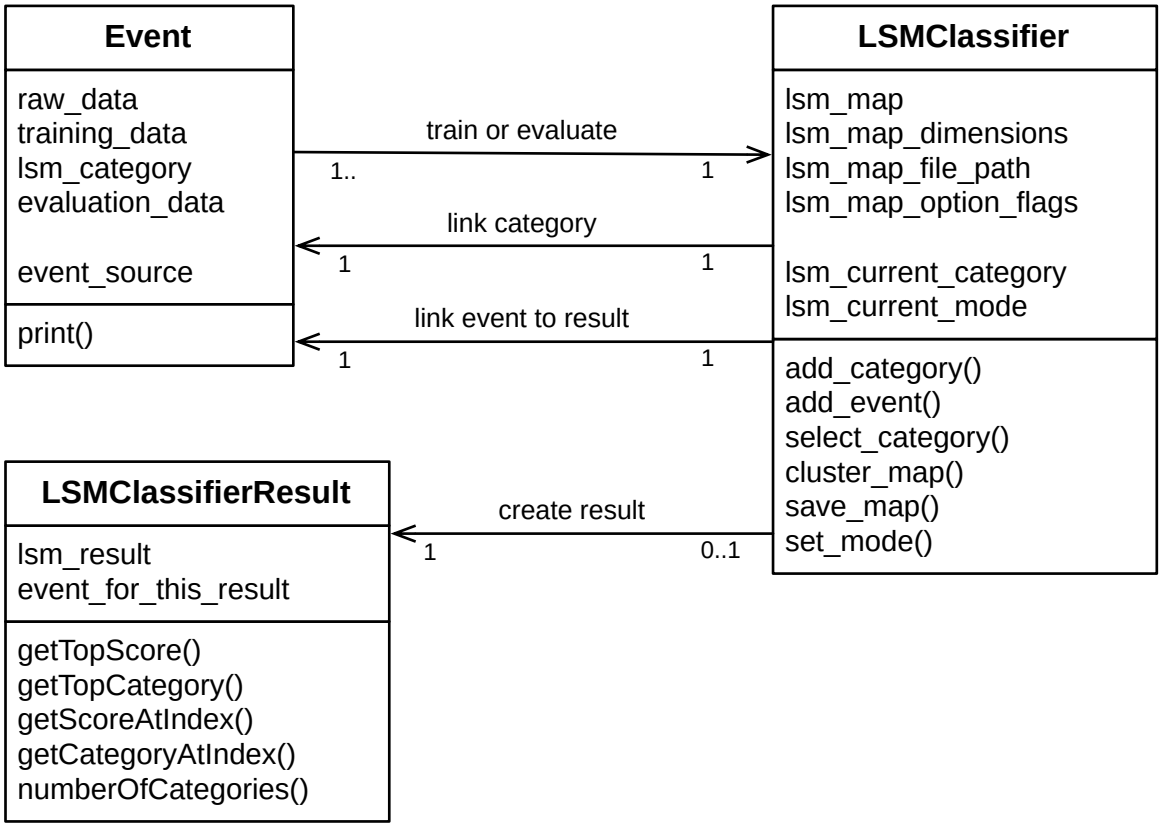

Figure 3.7: LSM Classifier Internals

3.3.7 Post-processing and Analysis

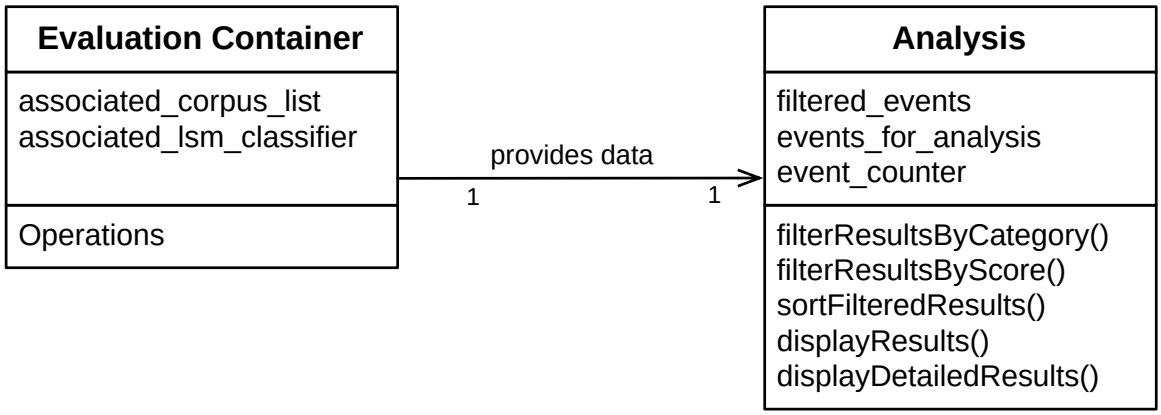

Figure 3.8: Analysis Class

Finally, the LSMPP software deals with the analysis and display of the result data. The analysis class responsible for this is depicted in Figure 3.8; it is used to filter, sort and display the results. Filtering may include actions such as excluding results that are under a certain LSM score, equating to less certainty in the context of the paradigm.

Additionally, summarisation and user output is handled by the analysis class: the output is formatted for post-processing and analysis by 3rd party tools such as Excel. The final analysis is done by the user based on the output of the LSMPP software.

3.4 Use-Case Detail

Identification, categorisation and disambiguation of event data, as discussed in the Literature Review, are some of the most prevalent challenges that event specialists face in production environments. The quality of the event parsing has a large impact on the efficacy of a log management system or SIEM, as well as the level of confidence that can be bestowed upon reports and dashboards - often the only instrumentation available to information security managers and business owners.

The use-cases that follow in this chapter are selected to test LSM in scenarios likely to occur within a security monitoring practice or that practitioners and analysts are likely to encounter when engaging in forensics and incident response.

Table 3.1: Use-Case Index

Use-Case	Test	Location	Deals With
1	Detect a Trained Eventtype	section 3.5	Identification
2	Detect Multiple Sourcetypes from a Single Stream	section 3.6	Classification
3	Detect Different Sourcetypes and Eventtypes Using Clustering	section 3.7	Disambiguation

Table 3.1 shows the use-cases that are addressed in this research. These use-cases cover the core functionality of LSM and can lead to many derivative use-cases. A listing of derivative use-cases can be found in Appendix A, Table A.1.

3.5 Use-Case 1: Detect a Trained Eventtype

A common challenge faced by SIEM implementers is assuring that they have identified all the possible variations of an eventtype occurring in the system. In order for correlation rules, such as detecting brute-force logon attempts, to function properly, the SIEM system needs to be made aware of all the relevant logon events. This may seem simple but in practice it can often prove to be quite challenging.

The aim of this experiment was to see if LSM can assist the SIEM implementer in identifying *all*successful logon events within the test data set.

To accomplish this goal, nineteen "logon success" events from different sourcetypes were used to train the LSM classifier. These events were split into four distinct, randomly selected groups and evaluated against a collection of 1000 test events. A successful run would then return similar or closely related events.

In the first variation of this experiment, an attempt was made to train only successful logon events and run all tests. Unfortunately, LSM does not allow for single-category evaluation and this attempt was consequently unsuccessful. As discussed in subsection 2.3.2, the LSM API provides a weighted list of categories with associated scores. The scores from the results always sum to 1.0 and as such, single category maps always return a 1.0 weighted result score to the provided evaluation data.

In the second variation of the experiment, a collection of counter samples was provided to the training phase. This counter sample collection was created by randomly selecting log events that were not successful logon events.

3.5.1 Design

The process followed in this use-case is described in diagram Figure 3.9. The diagram illustrates the distinct processing phases which were followed, discussed in more detail in the section that follows. This section opens with a description of the Training Events and Counter Samples followed by a detailed description of the pre-processing phase and its use of the "AdvancedRegexStrategy" pre-processing mechanism. The chapter then continues with a detailed look at the steps followed for training and preparing the LSM classifier.

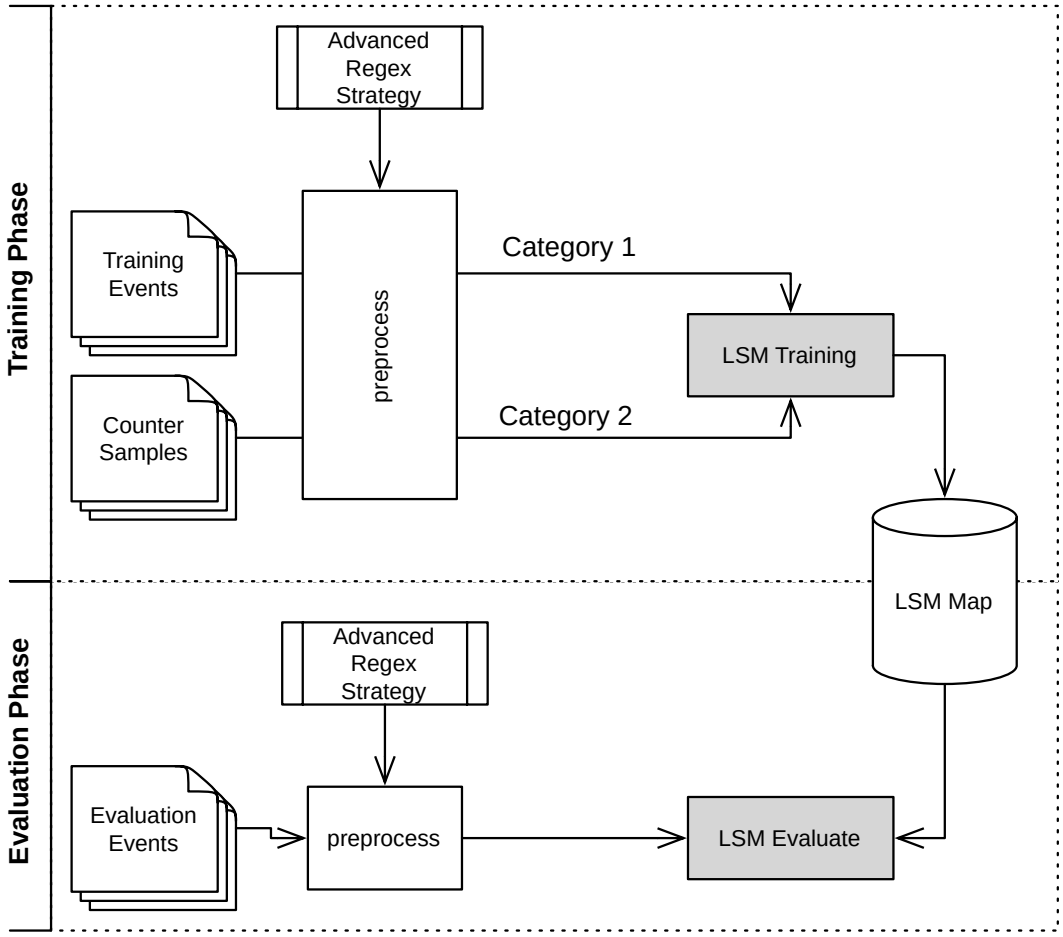

Figure 3.9: Use-Case 1 High Level Process Flowchart

The first processing step was the preparation of the relevant Training Events and Counter Samples. The training data collection contained nineteen logon events from different sources that were split into four different sets as shown in Table 3.2. This data was collected and prepared manually into individual files based on searches conducted on the data obtained for the research. Search terms such as 'logon', 'login' and 'authenticate'

led to relevant results after which the training data was saved to the files described in the sample data table. An example of one of the events can be seen in Figure 3.10.

```
Sep 18 09:09:01 ncorpnode1 EvntSLog: Fri Sep 18 09:09:01 2009
12221022 NCORPNODE1    Success Audit 528 Security Security
Logon/Logoff ACME\dxadmin
Successful Logon:
User Name: dxadmin
Domain:  ACME    Logon ID:(0x0,0xFCB92BD2)    Logon Type: 10
Logon Process: User32 Authentication Package: Negotiate
Workstation Name: NCORPNODE1
Logon GUID: {a4ec27ab-7aa5-8349-d751-7d987ee58a7f}
Caller User Name: NCORPNODE1$   Caller Domain: ACME
Caller Logon ID: (0x0,0x3E7)    Caller Process ID: 25436
Transited Services: -
Source Network Address: 192.168.0.1 Source Port: 52491
```

Figure 3.10: 528_ntsyslog_logon_success.txt

Counter samples were prepared by skimming through the research data set and selecting nineteen events that were not successful logon events.

The first phase is illustrated in diagram Figure 3.9 The training data events were loaded into memory after which pre-processing was applied to every loaded event. A detailed description of the pre-processing steps follows.

Table 3.2: Use-Case 1 Training Events

Set	Filename	Description
1	4624_logon_winsec_success.txt	A successful logon event to a Windows 2008 domain controller using the Kerberos protocol. The event code for this event is 4624
	528_ntsyslog_logon_success.txt	A successful logon event from a Windows NT domain controller. The event code for this event is 528
	528_winsec_auth_success.txt	A successful logon event from a Windows 2000 or 2003 domain controller. The event code for this event is 528
	540_ntsyslog_logon_success.txt	A successful network logon event from Windows NT. The event code for this event is 540
	dovecot_logon_success.txt	A successful login to a Dovecot IMAP server
2	f5_firepass_syslog.txt	A logon event from an F5 device collected using syslog
	fortinet_auth_success.txt	A successful login event to a Fortinet device for the establishedment of a SSL VPN tunnel
	kerberos_auth_success_winsec.txt	A successful network logon event from a Windows 2003 domain controller. The event code for this event is 540
	mac_osx_syspref_auth_success.txt	A successful authorisation for a system process on a Mac OSX system
	ossec_dovecot_auth_success.txt	A successful Dovecot IMAP login event collected using OSSEC
3	ossec_pam_unix_logon_success.txt	A successful login to a Unix system collected using OSSEC
	ossec_syslog_ftp_success.txt	A successful login to a FTP server collected using OSSEC
	pam_auth_su_seccess.txt	An event reflecting successful authentication using PAM
	interactive_osx_auth_success.txt	A successful login to a SSH session
	postgresql_success.txt	An reflecting successful authorisation of a connection to a PostgreSQL server
4	screensharing_auth_success.txt	A successful logon to a screen sharing session on a Mac OSX system
	ssh_key_success_logon.txt	A successful logon to ssh using public and private keys
	su_to_root_successfull.txt	Successful su by a user
	S540_ntsyslog_logon_success.txt	A successful logon to Windows NT Domain collected using Snare

3.5.2 The Advanced Regex Strategy

During the research, various pre-processing strategies were considered as discussed in section 3.3. For the purpose of the final experimental evaluations, a strategy named "AdvancedRegexStrategy" was used as it is a strategy involving more than 30 individual regular expression substitutions to produce a tokenised, semantically representative version of the event data. The strategy is illustrated in Figure 3.12.

The process initiates with a raw event of which dovecot_logon_success.txt (Figure 3.11) is an example. The event proceeds through several regular expression substitutions illustrated in the diagram in Figure 3.12 to produce the event illustrated in Figure 3.13. Each step in the diagram Figure 3.12 illustrates the starting state of the event as well as its resulting substitutions, highlighted in bold text, after regular expression substitutions were applied.

```
Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<dxadmin>, method=PLAIN,
rip=192.168.2.10, lip=192.168.1.10, TLS
```

Figure 3.11: dovecot_logon_success.txt

This first set of regular expressions, "Date and Time Pre-processing" in Figure 3.12, dealt with timestamps. Extensive effort was spent on identifying a wide variety of date and time formats prevalent in the training and evaluation data. Although not guaranteed to be exhaustive, the resulting regular expressions were able to successfully substitute all date and time strings in the data that were identified. All detected timestamps were replaced with dateToken tokens.

Examples of the various timestamp formats are shown in Table 3.3.

The step "IP Address Pre-processing" involved removing all detected IP addresses in the event and substituting an ipv4Token token.

The step "Other Pre-processing" included functionality to substitute email addresses, urls, fully qualified domain names, Microsoft Windows SIDs, numbers and common synonyms. Synonyms include the substitution off all occurrences of the word *login* with *logon* in order for the LSM classifier to only deal with the word 'logon'. Words such as 'successful' and 'succeed' were substituted with 'success'.

The reader can witness a false positive match in the second to last phase of the pre-processor where the string 'imap-login' gets replaced with a *suspectNameToken* token. A

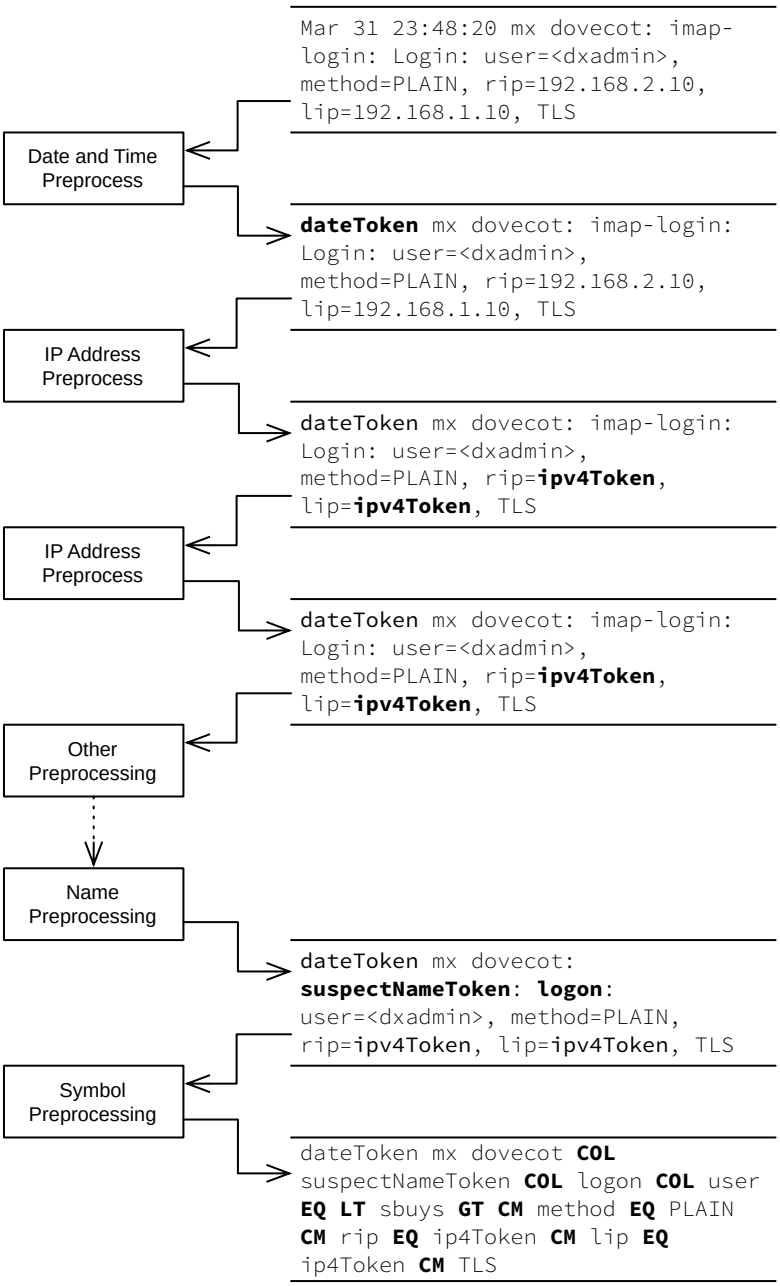

Figure 3.12: AdvancedRegexStrategy

```
dateToken mx dovecot COL suspectNameToken COL logon COL user EQ LT sbuys GT
CM method EQ PLAIN CM rip EQ ip4Token CM lip EQ ip4Token CM TLS
```

Figure 3.13: Preprocessed Event

Table 3.3: Timestamp Formats

Timestamp	Source
Feb 28 15:56:24	Syslog
date=2009–10–29 time=12:26:09	Fortigate
Mon_Jun__1_18:10:01_PDT_2009	Unidentified
2009–06–01 17:44:15.453	Unidentified
[06/10/2012 22:19:13.051]	ZenWorks
Fri Sep 18 09:09:01 2009	Windows

regular expression substitution was introduced to detect logon and machine names of the format 'john-workstation'; a decision was made to proceed with the evaluation regardless of this error as it was deemed un-noteworthy in terms of significant semantic difference to the data for this use-case.

3.5.3 LSM Classifier

All pre-processed training and counter sample data events were loaded into memory and then passed to the LSM classifier. All Training Events were routed to category 1 of the LSM Map and all Counter samples were routed to category 2.

The LSM Classifier internal workflow is illustrated in Figure 3.14. The section that follows guides the reader through the process.

During the map creation phase, the developer can set up to three "LSM Map Option Flags" in order to control the settings of the LSM map that would be created. These flags determine whether or not the map should generate bi-grams or tri-grams and whether or not the contents of the map should be hashed. Hashing is used for maps such as the LSM map in the Mac OSX unsolicited email filtering service in order to hide the contents of the map from casual observers at it may contain potentially offensive words that regularly occur in unsolicited email.

For the purposes of this use-case and for the research in general, it was appropriate to instruct the classifier to construct bi-grams. As this use-case is dealing with the "logon success" eventtype, bi-grams are illustrated in Table 3.4. Tri-grams are analogous to bi-grams and consist of a similar grouping, but consisting of three as opposed to two words. Tri-grams are illustrated in Table 3.5 and were not used in the final experiments

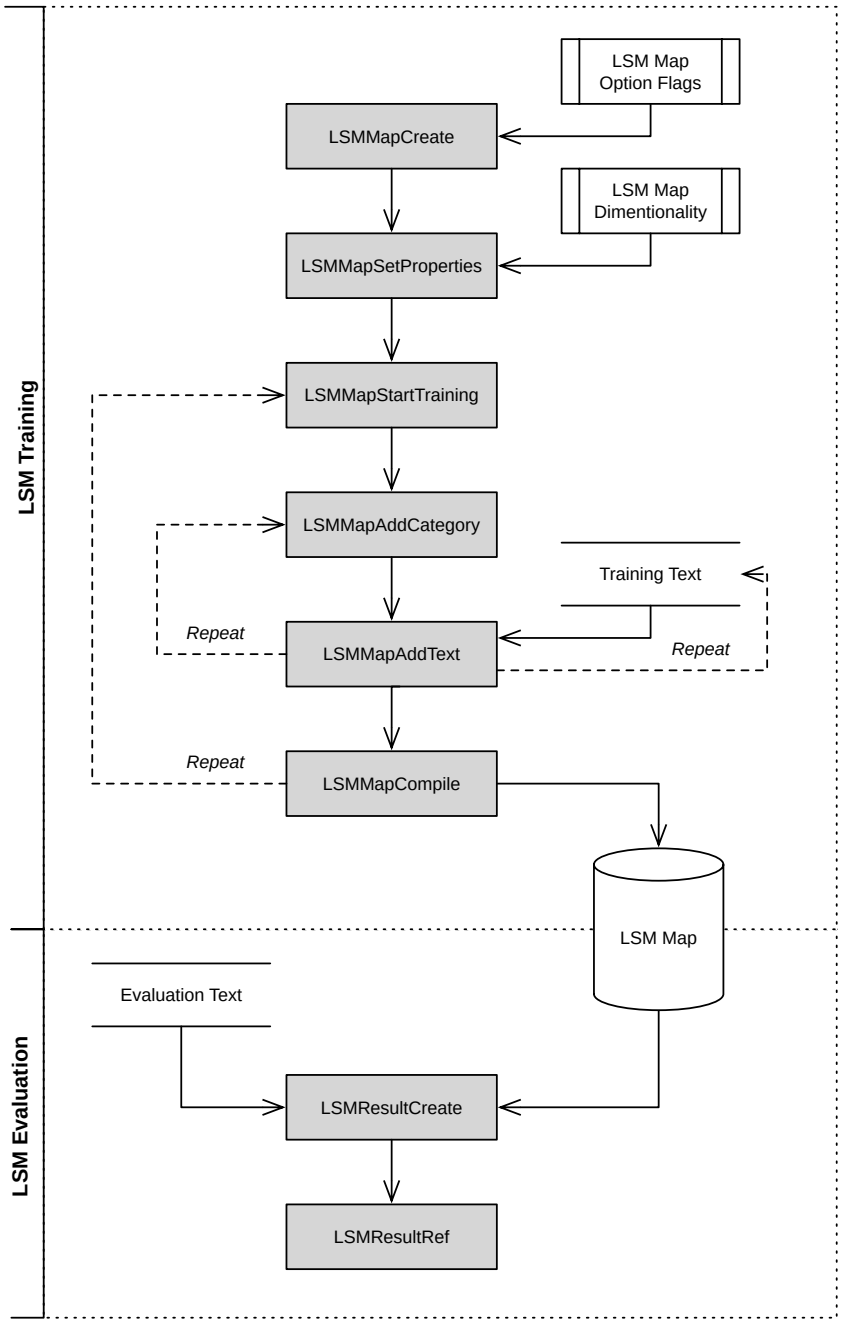

Figure 3.14: LSM Classifier Internals

of the research as the results obtained from tri-gram-based training did not appear to substantially affect the results during early experimentation. Furthermore, the results obtained from bi-gram training were sufficient to meet the goals of the research.

Table 3.4: Bi-grams

Preprocessed Event	Bi-Grams
User root logon success	user:root root:logon logon:success
Audit Success Event logon failed for user root	audit:success success:event event:logon logon:failed ...

Table 3.5: Tri-grams

Preprocessed Event	Tri-Grams
User root logon success	user:root:logon root:logon:success logon:success
Audit Success Event logon failed for user root	audit:success:event success:event:logon event:logon:failed ...

As discussed in section 3.3, events are treated as special types of documents, whereas with natural language processing the 'bag of words' model may suffice for many use-cases. Log events are more sensitive to word order and thus make it important to disambiguate between the bi-grams "logon:success" and "logon:failed".

After setting the appropriate flags, the LSM map is created using the *LSMMapCreate()* API call. The map is then explicitly set to "LSM Map Dimensionality" of 200, as illustrated in LSM Classifier Internals (Figure 3.14), using the *LSMMapSetProperties()* call. This increases the compilation performance and is sufficient when dealing with words mainly from the English dictionary (Kim *et al.*, 2003). Choosing the correct dimensionality is a topic that warrants its own research and thus is noted in the conclusions of this research as possible future research to assist in further refinements of the LSM classification performance. Sufficient accuracy was achieved using the chosen dimensionality to support the conclusions within the research.

The LSM classifier operates in two states as mandated by the API: the training and the evaluation states. After map creation, the classifier is set to the training state using the API call *LSMStartTraining()*.

Before any training data can be added to the LSM classifier, a category for that data needs to be created; for this use-case, an initial category was created for all of the training data

using the *LSMMapAddCategory()* call. Categories are internally denoted as integer values starting at 1.

After the category is created, the classifier can accept training data. The pre-processed training data gets added to the classifier at this stage; for this particular use-case, this resulted in between four to nineteen calls to *LSMMapAddText()*, depending on the training data set being evaluated.

LSMMapAddCategory() is then called to create category 2, after which the pre-processed counter sample data is added to the classifier. The diagram shown in Figure 3.14 illustrates that the above *LSMMapAddCategory()* and *LSMMapAddText()* calls may be repeated multiple times as needed.

The process concludes with a call to *LSMMapCompile()* which initiates SVD of the sparse matrix of the training data and results in an LSM map which can be saved to disk and which is now in the evaluation state.

3.5.4 LSM Evaluation

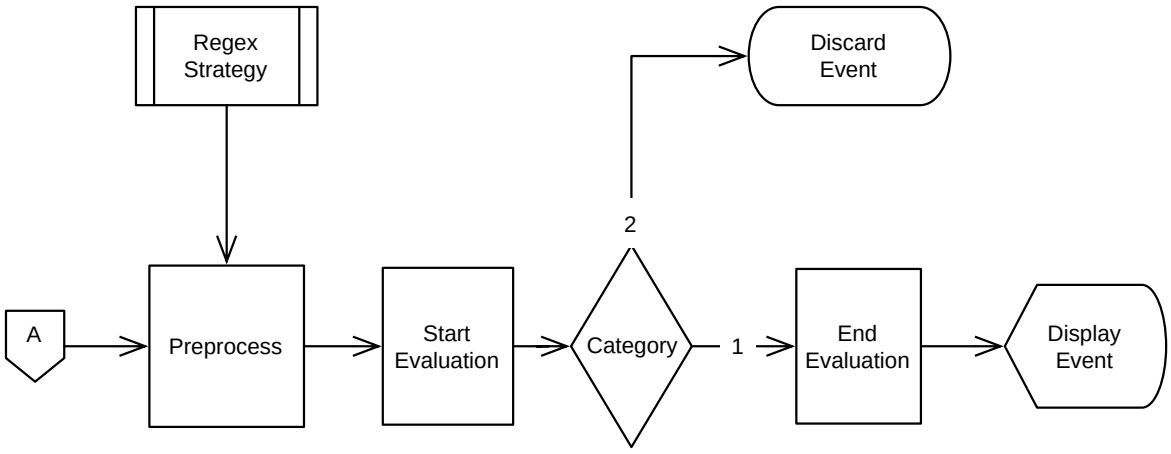

Figure 3.15: LSM Evaluation Phase

The evaluation phase is depicted in more detail in Figure 3.15. During the evaluation phase, all of the evaluation data gets loaded and pre-processed with the "AdvancedRegexStrategy" pre-processor as with the training data.

During this phase, the LSM classifier needs to have been set to its evaluation state using the call *LSMMapCompile()*. This call is essential and could result in "out of state" errors in the code if not set appropriately.

The code then loops through each of the evaluation events and a call to *LSMResultCreate()* then results in the creation of an *LSMResultsRef* object which is returned from the function. The result object is queried for its associated category using the *LSMResultGetCategory()* call.

All events that report a closest match to category 1 are then displayed along with their original raw content, and the associated LSM score is obtained through a call to *LSMResultGetScore()*.

3.5.5 Execution Details

Five distinct evaluation runs were programmed and executed, runs which were meant to test the following when training with a certain set of events:

- that the same kinds of events would be returned in kind, or
- that additional logon success events not in the training set would be returned.

This enabled the research to test if a SIEM implementer can train an LSM classifier with certain kinds of eventtypes and have LSM return similar events, as well as events of the same eventtype that were not included in the training data. In theory, this should have been possible due to the co-occurrence of words such as 'success' and 'logon' in a wide variety of 'logon success' events.

For this use-case, a dimensionality of 200 and bi-gram creation was chosen for all generated LSM maps. The same set of counter samples was used for all five evaluation runs.

A training set of 1000 randomly selected events was saved into a file and used for all of the evaluation runs; additionally, a Dovecot Logon similar in structure to Figure 3.11 was manually added to line 100 of the test set for verification purposes. Each evaluation run should at least have returned this event.

3.5.6 Anticipated Findings

It was anticipated that LSM would perform well with this use-case as it is analogous to the unsolicited email filtering use-case, but it was understood up-front that certain

properties of events could affect the outcome of the experiment. The major concern was the relative brevity of events when compared to most email; also, there was a concern over the availability of sufficient training samples.

3.5.7 Results

Table 3.6: Number of Results

Set	Count
1	13
2	127
3	133
4	237
Combined	157

Table 3.6 summarises the results of all of the evaluation runs. As can be seen, the number of results varied drastically depending on the events included in the training data. The results were the events from the evaluation data that matched category 1 and were displayed to the user.

Results for Set 1

The results of the evaluation against the first set of data is shown in Table 3.7; as discussed, only results from category 1 are displayed, events were shortened to 120 characters and relevant words were highlighted.

When trained with the first set of training events, Set 1 in Table 3.2 the test event "Dovecot Login" (no. 1) as well as two additional kinds of logon-related events were returned. Of the two logon eventtypes, only one event (no. 2) was of the "logon success" eventtype. Events 4 to 7 were all of eventtype "logon failure" and had a similar structure, after which various other eventtypes were flagged as being in category 1. It can be seen that the related LSM score, which is the distance measure of the evaluation data to the semantic anchor of category 1 in the LSM n-dimensional space, decreases steadily as the result quality decreases.

Of particular interest in this set was that only thirteen results from 1000 events in the evaluation data were returned, and that, other than the "Dovecot Login" seeded event, all the results were Microsoft Event Log events. This reflects the fact that Set 1 Table 3.2

Table 3.7: Set 1 Results Summary

no.	score	event
1	0.654984	Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<sbuys>, method=PLAIN, rip=192.168.0.1, lip=192.168.0.1, TLS
2	0.650625	Audit Success,2012/06/07 12:25:28PM,Microsoft-Windows-Security-Auditing,4624,Logon,"An account was successfully logged...
4	0.641888	Audit Success,2012/06/07 12:30:55 PM,Microsoft-Windows-Security-Auditing,4634,Logoff,"An account was logged off. Subject...
6	0.641888	Audit Success,2012/06/07 12:18:25 PM,Microsoft-Windows-Security-Auditing,4634,Logoff,"An account was logged off. Subject...
7	0.639554	...,4634,Logoff,"An account was logged off. Subject: Security ID: **DOMAIN\username** Account Name: **username** Ac...
8	0.585295	... Auditing,5447,Other Policy Change Events,"A Windows Filtering Platform filter has been changed. Subject: Security ...
9	0.571769	Warning,2012/05/24 08:05:24 AM,Microsoft-Windows-Security-Licensing-SLC,12321,None,Token-based Activation failed.
10	0.567427	... Auditing,5447,Other Policy Change Events,"A Windows Filtering Platform filter has been changed. Subject: Security...
13	0.538610	... Auditing,4932,Directory Service Replication,"Synchronization of a replica of an Active Directory naming context has ...

also contained four Microsoft Event log training samples. All of the "logon failure" events were of event code 4634 (An account was logged off[6]).The the difference of LSM scores for these events reflects that events 4 to 6 were SYSTEM user logoffs, whereas event 7 was a domain user logoff.

It was encouraging that the logon events all received the highest LSM scores and that there was a clear correlation between the drop-off in relevancy of the results and the related LSM scores. The lowest ranking was a Microsoft Active Directory Replication event and was only related to the training data due to the structure that Windows Event Log events share, such as often repeated words (Audit Success, Microsoft, etc).

Results for Set 2

The results for the second set of training events were reflected in Table 3.8. As with the previous results, the table contains a selection of relevant results. Events that were discussed were highlighted and all events were truncated where necessary.

The training data for Set 2 consisted of events from F5 firepass, Fortinet, Windows Event Log, Mac OSX as well as a Dovecot Auth Success event, this time in the form of an OSSEC event as reflected in Set 2 in Table 3.2.

The number of results returned from this training set's evaluation, 127, differs drastically from the thirteen results returned for Set 1 as reflected in Table 3.8. Additionally, if the criteria for this use-case are relaxed from "logon success" events to just "logon" events, then this evaluation is a major success with only 6 of the 127 (5%) events being false positives. The false positive results are all ranked lower than the "logon" events when the results are sorted according to LSM score.

As with Set 1, this illustrated the ability of LSM to provide helpful insights into the evaluation data with relatively little effort.

In Set 2, many variations of logon failure events are from a common source, Secure Shell (ssh). Event 110 and 123 illustrate the difference in event structure when a user exists versus when a user does not exist on a target system. This information is valuable when detecting brute-force user list attacks versus brute-force password cracking attacks.

[6]Description of security events in Windows Vista and in Windows Server 2008, http://support.microsoft.com/kb/947226

Table 3.8: Set 2 Results Summary

no.	score	event
1	0.646625	Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<sbuys>, method=PLAIN, rip=192.168.0.1, lip=192.168.0.1, TLS
84	0.596312	Apr 17 15:28:39 acmefileserver sshd[7734]: Invalid user teacher from 192.168.0.1
91	0.593993	Audit Success,2012/06/07 12:30:55 PM,Microsoft-Windows-Security-Auditing,4634,Logoff,"An account was logged off...
94	0.583235	Audit Success,2012/06/07 12:21:37 PM,Microsoft-Windows-Security-Auditing,4624,Logon,"An account was successfully logged on...
95	0.581767	Warning,2012/05/24 08:05:24 AM,Microsoft-Windows-Security-Licensing-SLC,12321,None,Token-based Activation failed.
102	0.571452	Apr 17 12:17:01 acmefileserver CRON[3535]: pam_unix(cron:session): session opened for user root by (uid=0)
104	0.570035	... Auditing,5447,Other Policy Change Events,"A Windows Filtering Platform filter has been changed. Subject: Security...
106	0.562977	... Auditing,4932,Directory Service Replication,"Synchronization of a replica of an Active Directory naming context has begun.
107	0.561928	Apr 13 19:24:02 acmepayroll sshd[14139]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=DOMAIN user=root**
110	0.559026	Jun 4 14:27:27 dryder sshd[67006]: error: PAM: authentication error for root from 192.168.0.1
119	0.536631	Apr 14 15:16:52 acmepayroll sshd[10007]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.0.1
123	0.530100	Jun 4 14:42:35 kelvin sshd[56167]: error: PAM: authentication error for illegal user super from 192.168.0.1
124	0.526152	Apr 17 06:44:45 acmefileserver sshd[31656]: Failed password for invalid user update from 192.168.0.1 port 58175 ssh2
127	0.515225	Apr 12 12:27:30 acmepayroll sudo: root : TTY=pts/0 ; PWD=/etc/apache2/sites-enabled ; USER=root ; COMMAND=/usr/bin/vim convergence_wsgi

Additionally, messages 110, 119, 123 and 124 are all variations of the same eventtype ("logon failure") that differ in structure and wording. Compare the use of "illegal user" in event 123 versus the use of "invalid user" in event 123. Of importance to a SIEM implementer here is that extra care needs to be taken to assure that sufficient coverage of all "logon failure" events are achieved. In order to illustrate, compare the regular expressions in table Table 3.9.

Table 3.9: Regular Expressions

Label	Regular Expression	Line	Original Event	Username
A	authentication\sfailure;.+user=([^\s]+)\s	107	Apr 13 19:24:02 acmepayroll sshd[14139]: pam_unix(sshd:auth): **authentication** failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=DOMAIN **user**=root	root
B	authentication\serror\sfor\s([^\s]+)	110	Jun 4 14:27:27 dryder sshd[67006]: error: PAM: **authentication** error for root from 192.168.0.1	root
C	authentication\sfailure;.+ruser=([^\s]+)\s	119	Apr 14 15:16:52 acmepayroll sshd[10007]: pam_unix(sshd:auth): **authentication** failure; logname= uid=0 euid=0 tty=ssh **ruser**= rhost=192.168.0.1	-
D	illegal\suser\s([^\s]+)	123	Jun 4 14:42:35 kelvin sshd[56167]: error: PAM: **authentication** error for illegal **user** super from 192.168.0.1	super
E	invalid\suser\s([^\s]+)	124	Apr 17 06:44:45 acmefileserver sshd[31656]: Failed password for invalid **user** update from 192.168.0.1 port 58175 ssh2	update

In the event of the use of a correlation engine that employs regular expression matching, the SIEM implementer would have to define the five patterns defined in Table 3.9 to cover the five variations of "logon failure" events for the sourcetype "ssh" that we detected with the evaluation data used in the use-case. This clearly illustrates one significant challenge faced as well as the beneficial usefulness of LSM within this domain.

When comparing the four variations, only the events in row A, B, C and D from Table 3.9

contain the word 'authentication'; the event in row E uses the term 'failed password'. None of the events contains the words 'logon' or 'login'. Only A, D and E contain the word 'user'. A naive implementation trying to capture usernames may try and match any string following the word 'user'; that would, however, fail for events B and C as it would be susceptible to errors if attackers were to inject the word 'user' into the username field. For example, a match with event B may look like "error for *user* from"; a naive regular expression match would then detect the username for this user as the word 'from'.

The results from this set of training events are encouraging and have already led to several insights that the SIEM implementer may easily have missed.

Results for Set 3

Table 3.10: Set 3 Results Summary

no.	score	event
1	0.653261	Apr 17 12:17:01 acmefileserver CRON[3535]: pam_unix(cron:session): session opened for user root by (uid=0)
		...
103	0.638141	Apr 17 15:32:36 acmefileserver sshd[10077]: pam_unix(sshd:auth): check pass; user unknown
		...
133	0.515131	Apr 12 12:27:30 acmepayroll sudo: root : TTY=pts/0 ; PWD=/etc/apache2/sites-enabled ; **USER**=root ; COMMAND=/usr/bin/vim convergence_wsgi

The training data for Set 3 contained events collected using OSSEC as well as some more traditional ssh and PAM-based authentication success messages as shown in Set 3 in Table 3.2. The results for this set are dominated by ssh "logon failure" events, with only two logon success events.

An exciting find for this set of evaluations is event 133 which captured a successful 'sudo' command. This is a clear illustration of the ability of LSM to detect event variations that did not exist in the training data; as a matter of fact, there are no 'sudo' events in any of the four training sets. By investigating the event structure as produced by the pre-processor, the most likely reason that event 133 appeared in the results is that it contained the sequence "user EQ" which resulted in the bi-gram "user:EQ" and also appeared in the prostgresql_success.txt training data that was included in this training

run. The associated score for this match is 0.515131 which implies, when compared to the strongest match event 1, 0.653261, that it was a relatively weak match and could easily have been missed had any filtering on strength of the matches been in place.

A new variation of ssh logon failure events is also brought to light with event 103 and similar messages to it.

Disappointingly, the Dovecot Auth Success seeded into the evaluation data test event did not get classified into category 1 and because of this, was not displayed as a part of the results for Set 3 even though it contains the word 'login' and the word sequence 'user EQ'. One hundred percent of the events in this result set are 'logon' events, a result not reflected in the other evaluation runs and indicative of the relative terseness of the results in the training data.

Results for Set 4

Table 3.11: Set 4 Results Summary

no.	score	event
1	0.638299	Apr 17 15:34:59 acmefileserver sshd[11228]: Failed password for invalid user admin from 192.168.0.1 port 56528 ssh2
129	0.599685	Apr 17 12:17:01 acmefileserver CRON[3535]: pam_unix(cron:session): session opened for user root by (uid=0)
133	0.595744	Audit Success,2012/06/07 12:18:25 PM,Microsoft-Windows-Security-Auditing,4634,Logoff,"An account was logged off...
134	0.594867	Audit Success,2012/06/07 12:25:28 PM,Microsoft-Windows-Security-Auditing,4624,Logon,"An account was successfully logged on...
136	0.584847	Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<sbuys>, method=PLAIN, rip=192.168.0.1, lip=192.168.0.1, TLS
139	0.574475	Apr 16 08:41:13 acmefileserver sshd[16406]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=DOMAIN
140	0.568002	May 26 09:41:53 dryder ntpd[800]: kernel time sync status change 6001
237	0.501776	Jul 31 13:30:27 192.168.0.1 [DHCP IP: (192.168.0.1)] to MAC address CC:08:E0:12:37:5D,

The composition of Set 4 of the training events is shown in Table 3.2. It contained events from the OSX screen sharing service, key-based successful authentication to ssh,

the issuance of the su command as well as a Windows successful logon from a Windows NT System. The results are again largely dominated by ssh events; this is not surprising based on previous result sets.

Of interest in this set, though, is that all of the top 139 events shown in Table 3.11 are "logon" events and as such the classifier performs very well. A simple sort of the events by evaluation score, as with all of the results, clearly delineates the true positive from the false positive results: the number of false positive events in this run approaches 40% of the total set of results which numbered 237 events.

Windows Event Log events again appear in the results; this can be attributed to the inclusion of a Windows NT logon success event in the training data. An encouraging finding is that the training data contains an event code 540 successful logon event, but that the evaluation data contains events (133 and 134) from Windows 2008 servers, and the returned events are of event code 4634 and 4624. This bodes well for the use of LSM where systems may periodically be upgraded and where the syntactic structure of events may change, even though the event contents remain mostly similar.

Results for Combined Set

With the final run, all of the training data from Table 3.2 were combined. The 157 results produced contained most of the results from the previous sets with one notable exception: the 'sudo' command detected in the results of Set 3.

As with the previous runs, the "logon" events had a higher LSM score than most other of the other events, the remaining 10%, and as such, the events are clearly partitioned for the end user. The results contained roughly 10% fewer false positives than Set 4. It is summarised in Table 3.12.

Table 3.12: Combined Run Results Summary

no.	score	event
1	0.648782	Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<sbuys>, method=PLAIN, rip=192.168.0.1, lip=192.168.0.1, TLS
2	0.648370	Audit Success,2012/06/07 12:25:28 PM,Microsoft-Windows-Security-Auditing,4624,Logon,"An account was successfully logged on...
8	0.616020	Jun 4 07:14:11 dryder sshd[34217]: error: PAM: authentication error for illegal user security from 192.168.0.1
90	0.610023	Apr 17 12:17:01 acmefileserver CRON[3535]: pam_unix(cron:session): session opened for user root by (uid=0)
91	0.609866	Apr 14 21:48:55 acmepayroll sshd[1827]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.0.1 user=root
96	0.608430	Apr 17 15:34:59 acmefileserver sshd[11228]: Failed password for invalid user admin from 192.168.0.1 port 56528 ssh2
103	0.596225	Jun 4 14:27:27 dryder sshd[67006]: error: PAM: authentication error for root from 192.168.0.1
105	0.592346	Apr 17 15:28:39 acmefileserver sshd[7734]: Invalid user teacher from 192.168.0.1
112	0.590960	Jun 4 13:52:56 dryder sshd[64455]: error: PAM: authentication error for news from 192.168.0.1
114	0.587433	Apr 14 22:07:43 acmepayroll sshd[8376]: Failed password for root from 192.168.0.1 port 37444 ssh2
125	0.584880	Apr 17 15:32:36 acmefileserver sshd[10077]: pam_unix(sshd:auth): check pass; user unknown
134	0.575172	Apr 17 15:35:00 acmefileserver sshd[11243]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.0.1
145	0.528018	Audit Success,2012/06/07 12:32:41 PM,Microsoft-Windows-Security-Auditing,4932,Directory Service Replication...
157	0.501826	Information,2012/05/28 04:25:56 PM,Microsoft-Windows-MSDTC 2,4202,TM,"MSDTC started with the following settings...

3.5.8 Findings

Table 3.13: Results Summary for Use-Case 1

	Set 1	Set 2	Set 3	Set 4	Combined
# Results	13	127	133	237	157
# Logon Success Events	2	4	2	4	4
# Logon Events	7	121	133	139	139
# False Positives	6	6	0	98	18
# False Negatives	127	13	7	0	0
Lowest Correct LSM Score	0.639554	0.515225	0.515131	0.574475	0.531616
Highest Incorrect LSM Score	0.585295	0.581767	-	0.568002	0.606832
Incorrect Mixed into Correct?	No	Yes	No	No	Yes

The experiments conducted in this use-case illustrated the potential use of LSM for SIEM implementers when having to deal with large sets of data, where the eventtype of the incoming data needs to be identified. This is useful for the SIEM implementer for detecting variations on the data that may have been unanticipated, for example a new format of an event after an upgrade to software, or during the implementation or normalisation phases, where a high degree of accuracy is needed for reporting and for correlation to truly be effective.

Table 3.13 summarises the finding for the this use-case. The impact of training data on the classifier's performance can be seen clearly in the difference in False Negative counts between Set 1, 4 and the Combined Set. Set 3 clearly had the most success clarity although, unfortunately, seven logon events were not classified.

It is common practice in information retrieval to measure the *recall* and *precision* of machine classification. Additionally, the *f-measure* of a classifier's performance gives the evaluators a combined metric (Bellegarda, 2008). *Recall* is a measure of the classifier's ability to identify all the relevant events. *Precision* is a measure of the classifier's accuracy, the ability to identify only relevant events. The *f-measure* (Riloff and Lehnert, 1994) is calculated to provide a summary of the overall performance of the classifier.

$$recall = \frac{number\ of\ relevant\ events\ retrieved}{total\ number\ of\ relevant\ events}$$

$$precision = \frac{number\ of\ relevant\ events\ retrieved}{total\ number\ of\ retrieved\ events}$$

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

Table 3.14: Use-Case 1 Classifier Performance

	Logon Success			Logon		
	Recall	Precision	f-measure	Recall	Precision	f-measure
Set 1	0.50	0.15	0.24	0.05	0.54	0.09
Set 2	1.00	0.03	0.06	0.87	0.95	0.91
Set 3	0.50	0.02	0.03	0.96	1.00	0.98
Set 4	1.00	0.02	0.03	1.00	0.59	0.74
Combined	1.00	0.03	0.05	1.00	0.89	0.94

Table 3.14 represents the performance of the classifier and as can be seen, the results vary wildly based on the training data provided. Be that as it may, the "logon" eventtype classification for Sets 2, 3 and Combined performed very well.

LSM is assumed to perform at its best when there are clear differences between the categories that need to be dealt with, as discussed in the Literature Review in section 2.3. In the tests, the structure of "logon failed" and "logon success" events did not vary enough and this was strongly reflected in the results seen in Table 3.14. The lesson learnt through this was that for broad categories of eventtype classes, LSM seems potentially more useful. LSM performed well when disambiguating both types of "logon" events from the counter samples.

It is noted that the use-case deals with a relatively simplistic approach to training data and counter sample selection and that results may be refined and improved with more iterations of the process. For example, if an explicit category for "logon failed" events were to be added to the classifier using the events from the results, the resulting classifier could be used to disambiguate the events even further. This process could be repeated or the classifiers could be serialised, with the original classifier processing events into "logon" and "not logon" events and a subsequent classifier disambiguating the events contained in the "logon" results through the use of clustering. Clustering is investigated in Use-Case 3 (section 3.7).

The experiment yielded valuable variations of different ssh events, as shown in the table below Table 3.15. This subtle difference has enormous value when dealing with correlation and analysis but is also extremely important to detect during the normalisation and classification processes of SIEM implementation.

The importance of correct training data selection is illustrated clearly by the difference in results with the four evaluation sets. It is a known property of LSM that its performance is affected by the training data provided, as previously discussed in section 2.3. It

Table 3.15: SSH Event Variations

SSH Event Variations
Jun 4 14:27:27 dryder sshd[67006]: error: PAM: authentication error for root from 192.168.0.1
Jun 4 14:05:30 dryder sshd[65588]: error: PAM: authentication error for *illegal* user uucp from 192.168.0.1
Jun 4 13:52:56 dryder sshd[64455]: error: PAM: authentication error for news from 192.168.0.1

was encouraging to find that the final evaluation, where all training data was combined, also delivered good results, implying that the challenge of training data selection can be managed.

The differences do, however, illustrate that the system relies heavily on a proper coverage of samples within the domain. Compare the number of results from Set 1 to the results obtained in the final evaluation. Set 1 was dominated by Windows Event Log events and as a result the bulk of the "logon" events detected using the other sets was not detected. For LSM to be effective in this use-case, an event specialist would need to have access to sufficient samples, either obtained through manual searching as with this research, or provided by a third party.

From the perspective of a SIEM implementer, this use-case is deemed to be a success if the original requirement is relaxed from detecting "logon success" to detecting "logon" events. Multiple iterations become practical if LSM is integrated properly into a practical framework for the user through which samples and training data can be easily manipulated with the availability of the correct pre-processing.

The tool-chain is very important when dealing with LSM in a practical setting. As was seen with the results, the LSM distance measure, or score, provided a guideline to the user, but selecting the correct thresholds may be a challenge. To that end, it would be more beneficial to have access to the LSM results in an integrated environment that would allow the user to sort results, re-run or refine results or perhaps allow for the adjustment of the score threshold interactively.

During the evaluation of Set 3, a 'sudo' event was detected that did not appear in the results of the other evaluation runs. This detection points to the need to have variability in training data selection coupled with careful iteration when using LSM in this use-case. Naively training all sample data and counter samples may displace some of the

possible fringe results that could be obtained and could warrant careful iteration, as well as multiple runs, during normal LSM usage.

3.6 Use-Case 2: Detect Multiple Sourcetypes from a Single Stream

Log sources can often blend different sourcetypes into a single event stream. A good example of this is a Syslog (Syslog, 2012) server that accepts Syslog messages from multiple types of source systems. The source systems may have no relation to each other or may be multiple instances of the same technology, producing the same sourcetype. It is often deceptively simple to put a Syslog server in place to facilitate log management. However, if such a system is not properly configured, all the different source systems may log their events into a common file such as /var/log/syslog.

When implementing a SIEM, it is often necessary to configure the system for specific sourcetypes. Support for different sourcetypes normally comes in the form of add-ons, also referred to as plug-ins or adaptors, which contain the relevant normalisations, regular expressions or other appropriate measures that enable the SIEM to recognise the appropriate fields. If these add-ons or plug-ins are misconfigured, or not enabled, the SIEM's functionality may be seriously hampered. LSM could potentially identify the relevant sourcetypes and help the SIEM implementer achieve a higher degree of confidence that all relevant add-ons have been enabled.

The aim of this experiment is to determine if a SIEM implementer or security analyst can use LSM to search for known sourcetypes in a large collection of events. The experiment should therefore use LSM to identify selected sourcetypes and report back to the user.

In order to facilitate this experiment, four sourcetypes were selected and an LSM classifier prepared with training data from the selected sourcetypes, which in turn were associated with four distinct categories within the LSM map. This map was used to evaluate a collection of 51,000 events randomly selected from the thesis data set. A successful evaluation would be able to indicate to the user that an evaluated event belonged to one of the pre-trained categories. Events that did not match any of the categories should be identifiable by a comparatively lower LSM score associated to the result.

3.6.1 Design

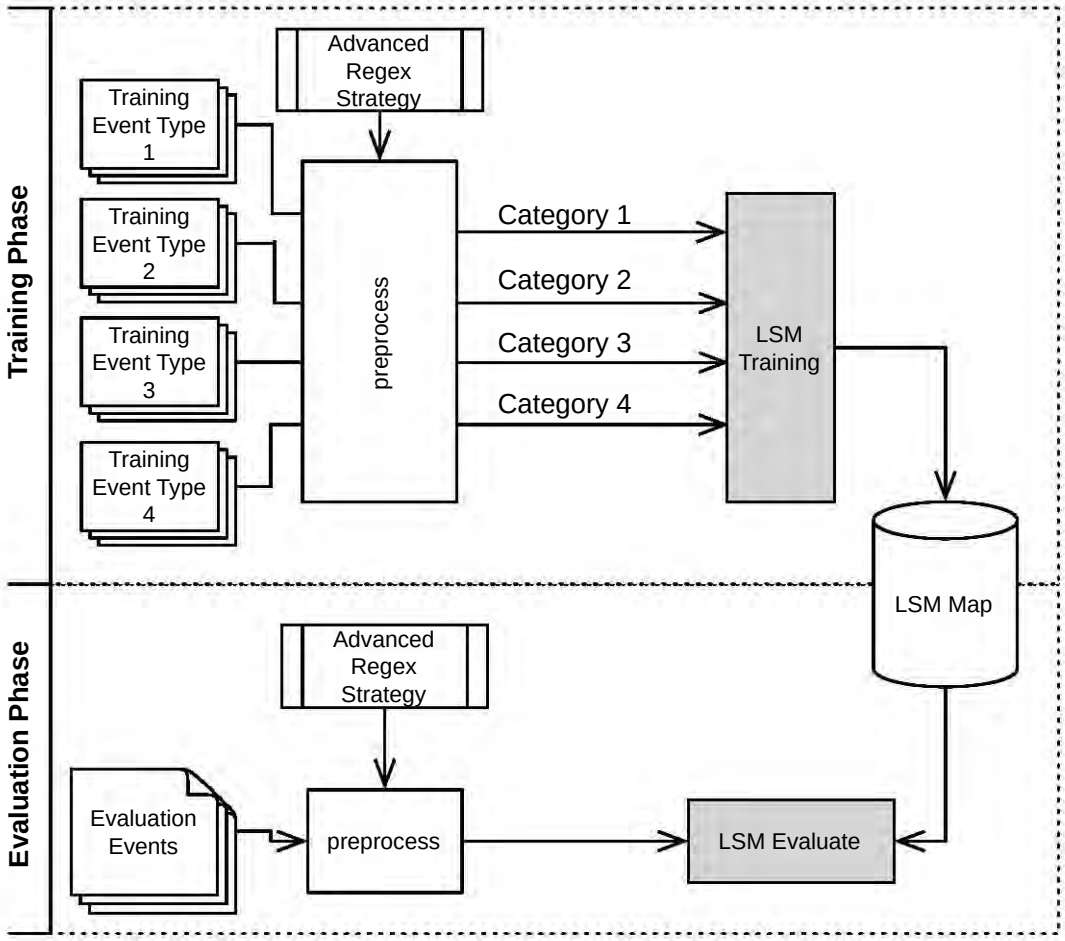

Figure 3.16: Use-Case 2 High Level Process Flowchart

This use-case shares several features with Use-Case 1 discussed in section 3.5, the major difference being that four different sourcetypes were chosen as opposed to a single event-type. In this experiment multiple eventtypes associated with a sourcetype were trained. Use-Case 1 dealt with detail related to the LSM processing framework and some of those details are not repeated within this section; the reader is referred to section 3.3 and section 3.5 for more detail related to the processes.

This section starts with the description of the training data used and then delves into some of the unique challenges faced when trying to analyse large data sets. It then concludes with the results of the experiments and the findings thereof.

Table 3.16 contains the category to sourcetype mapping for the training data. Little to no attention was given to the composition of the events within the training data files over

Table 3.16: Use-Case 2 Training Data

Category	Sourcetype	Description
1	Dovecot IMAP	A collection of 59 Dovecot events containing various eventtypes.
2	BIND	Nine BIND DNS server query events containing a single eventtype
3	F5	Eighteen F5 Application Security Manager events containing multiple eventtypes
4	Windows Security	Twenty Windows Security Event Log events containing multiple eventtypes

and above a check that it was the correct sourcetype and to reduce the count of the events in the training data files. This was done to accelerate the LSM training process as well as to simulate what an event specialist may do in a possible production environment.

As with Use-Case 1, a dimensionality of 200, suitable for the English language, was selected and the LSM classifier was instructed to create bi-grams. The process flow of the experiment is shown in Figure 3.16, where the primary difference to the previous use-case was the use of four LSM categories.

3.6.2 Anticipated Findings

It was anticipated that LSM would perform well as it was analogous to Use-Case 1. There was a concern that the larger set of training data used in comparison to Use-Case 1 could lower the quality of the results. The experiment, however, without undue attention given to the training data, continued as designed to stay true to the goal of this research which is to investigate if LSM could form a practical part of the tool-chain available to an event specialist. Having to train practitioners in the art of data composition and the nuances of training data selection seemed to be in conflict with this goal.

3.6.3 Results

In Use-Case 1, result analysis was relatively straightforward: results could either match or not match the trained eventtype and all results were sorted according to LSM score. The result sets were relatively small, not counting more than a couple of hundred. Use-Case 2, however, dealt with 51,000 results per experiment and is discussed in the next section.

As discussed in subsection 2.3.2, LSM result scores for an evaluation sum to a floating point number of 1.0. Results for all category 1 matches ranged from 0.273975 to 0.391778. An attempt was made to filter events according to LSM score but the threshold for such a filter was not apparent. Manual inspection and a couple of threshold setting test runs wielded no noteworthy results. An attempt to find a software-based programmatic solution led to a practical and informative visualisation of the results which proved to be sufficient for the experiment and pointed the way to possible user interface features in a production LSM tool, or alternatively, the basis for an algorithmic solution. The investigation of such a solution, however, was outside the scope of this research.

The graphs in the section that follows are based on a simple count of distinct LSM result scores due to the relative uniformity of the events after the pre-processing discussed in subsection 3.5.2; this approach worked surprisingly well. Most of the graphs clearly indicate large groupings of events and also the minimum and maximum LSM score ranges that the research should investigate. For completeness, the top results were also investigated for all experiments in this use-case to verify which results the classifier deemed to be most similar to the semantic anchor for that category.

Category 1 was trained with Dovecot IMAP training data and the results for category 1 matches are described in Table 3.17. The table contains selected events from the total result set, events which were selected for discussion based on a manual search for category 1 results and by seeking out the large clusters of positive matches visualised in Figure 3.17. The table illustrates a sampling of both false positive and false negative results returned from the classifier. It is encouraging that the true positives (event 46506 and 46509) ranked relatively high on an LSM score basis compared to other results but unfortunately, the visualisation did not assist in identifying these matches.

The true negative results (such as event 47856 and 49320) are disappointing as the DoveCot IMAP events were clearly misclassified. The events have been classified as Category 4 results, with the events to the table in the interest of providing comparative data. It is noted that *Dovecot Session Disconnected* events such as these did not appear in the training data and that may explain the phenomenon. The visualisation in Figure 3.17 did not assist with finding the true positive results and primarily just indicate that a large quantity of relatively weak results are contained in this result set due to the bias of larger counts towards the middle of the distribution.

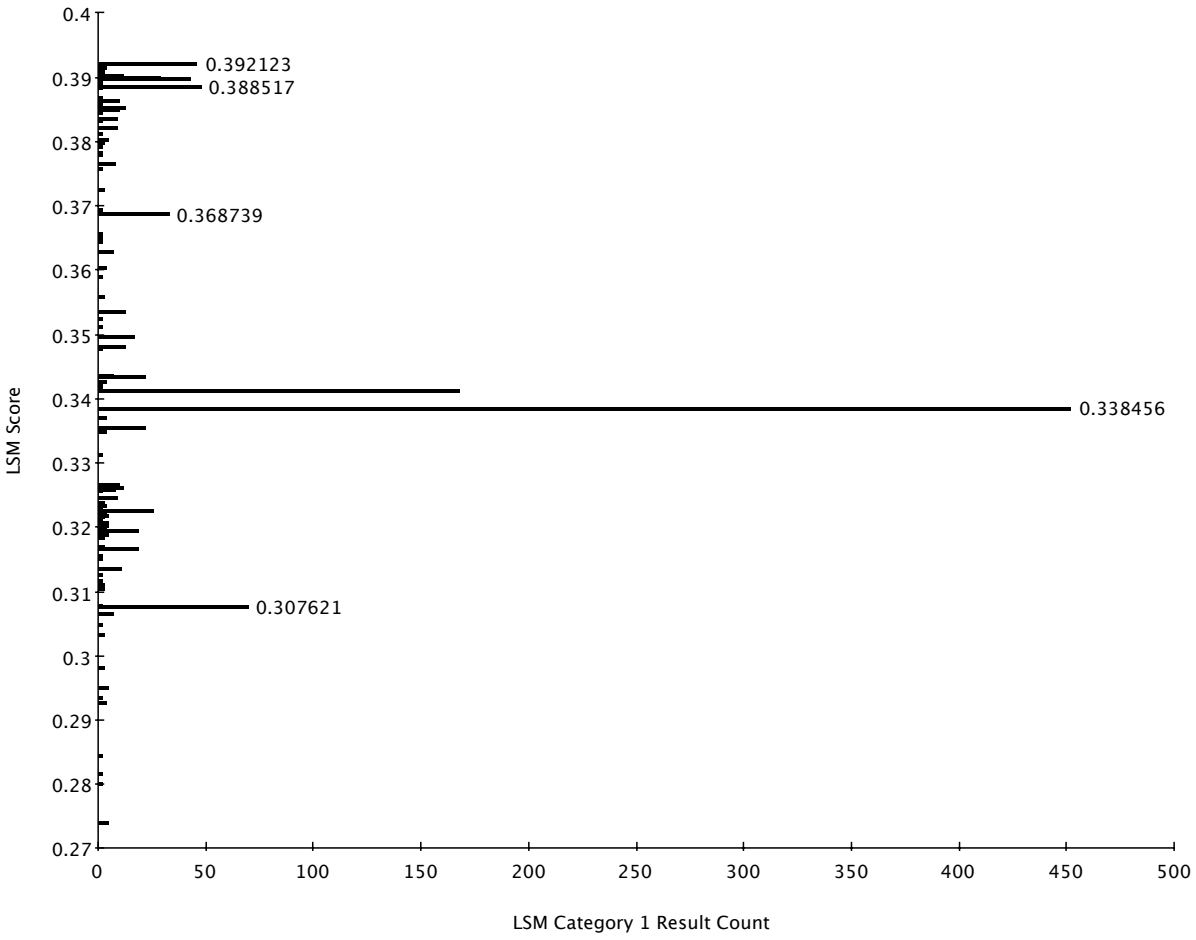

Figure 3.17: Use-Case 2: Selected Results (Category 1)

Table 3.17: Use-Case 2 Category 1 Results

no. result	category	score	event
59	1	0.392123	[DEBUG] [06/19/2012 12:30:48.428] [1700] [ZenworksWindowsService] [29] [] [CacheableHttpWebResponse] [] [Getting Response Stream] [] []
46191	1	0.388517	[DEBUG] [06/19/2012 22:30:55.960] [1700] [ZenworksWindowsService] [9] [] [SoapUtility] [] [Checksumming soap call with UID: Containment:GetContainment] [] []
46506 True Positive	1	0.379167	Feb 28 16:03:24 vmail dovecot: IMAP(user@DOMAIN): Disconnected
46509 True Positive	1	0.378285	Mar 31 23:48:20 mx dovecot: imap-login: Login: user=<sbuys>, method=PLAIN, rip=192.168.0.1, lip=192.168.0.1, TLS
46656	1	0.368739	Apr 15 21:47:34 192.168.0.1 [UPnP set event: Public_UPNP_C3] from source 192.168.0.1,
48869	1	0.338456	Jun 4 13:42:39 kelvin portfwd[13909]: copy: Failure reading from socket: Connection reset by peer
50699	1	0.307621	Jun 4 08:33:39 kelvin last message repeated 2 times
47856 True Negative	4	0.354017	Alert 1332871659.46765: - dovecot, 2012 Mar 27 20:07:39 mx->/var/log/syslog Rule: 9706 (level 3) -> 'Dovecot Session Disconnected.' Src IP: (none) User: (none) Mar 27 20:07:38 mx dovecot: IMAP(sbuys): Disconnected: Logged out bytes=477/7516
49320 True Negative	4	0.338442	Alert 1333174883.20854: - dovecot, 2012 Mar 31 08:21:23 mx->/var/log/syslog Rule: 9706 (level 3) -> 'Dovecot Session Disconnected.' Src IP: (none) User: (none) Mar 31 08:21:22 mx dovecot: IMAP(sbuys): Disconnected: Logged out bytes=257/1343

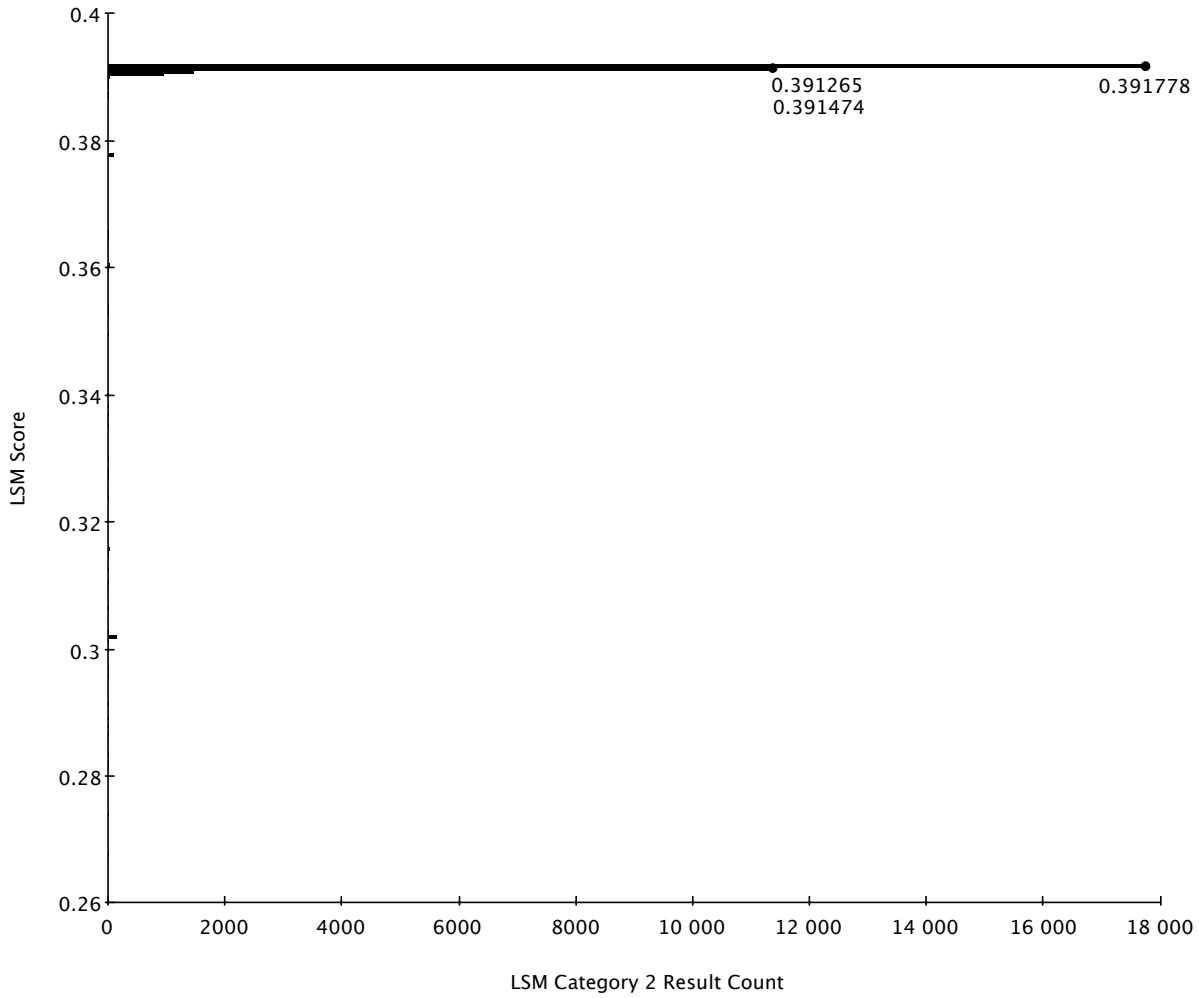

Figure 3.18: Use-Case 2: Selected Results (Category 2)

Category 2 is represented in Figure 3.18 and shows a large concentration of results at the very top of the LSM Score axis; results were investigated and summarised in Table 3.18. In Category 2, BIND DNS Query, matching proved to be very successful. It was also encouraging that the results were obtained from a relatively small collection of training events – nine events in factTable 3.16.

Of further interest is that Figure 3.18 is clearly dominated by results that have all clustered towards the maximum LSM result score obtained for this evaluation run. An investigation of the results shows that the top results are true positive results. False positive results rank lower than the true positive results and are relatively low in number as shown in Figure 3.18.

Table 3.18: Use-Case 2 Category 2 Results

no.	category	score	event
107	2	0.391778	16-May–2012 09:39:05.175 queries: info: client 192.168.0.1#25288: query: DOMAIN IN A -E
29233	2	0.391265	16-May–2012 09:38:48.586 queries: info: client 192.168.0.1#64418: query: DOMAIN IN AAAA -E
17858	2	0.391474	16-May–2012 09:35:36.099 queries: info: client 192.168.0.1#32819: query: DOMAIN IN A -
50999	2	0.267629	Information,2012/05/15 05:25:07 AM,SceCli,1704,None,Security policy in the Group policy objects has been applied successfully.

Category 3 deals with the attempted categorisation of F5 Application Security Manager (ASM) events. The evaluation data set, however, contained no F5 ASM Table 3.16 events and as such, no true positive results were anticipated for category 3. A summary of the results from the evaluation run can be seen in Table 3.19; the results are also visualised in Figure 3.19. The F5 ASM is a web application firewall that deals with web-related traffic. It is thus interesting to note that based on the samples from the results, results were dominated by firewall events and other content filtering solutions. Also, there were events from other F5 devices (event 47694 and 48214) which may be of some value to a SIEM implementer.

Of notable interest in this particular result was the distribution of events in the visualisation in Figure 3.19. The bulk of the events were not clearly clustered at the top of the LSM Score axis as with the results of category 2 Figure 3.18. As far as its distribution of events is concerned, the visualisation more closely relates to that of category 1 Figure 3.17.

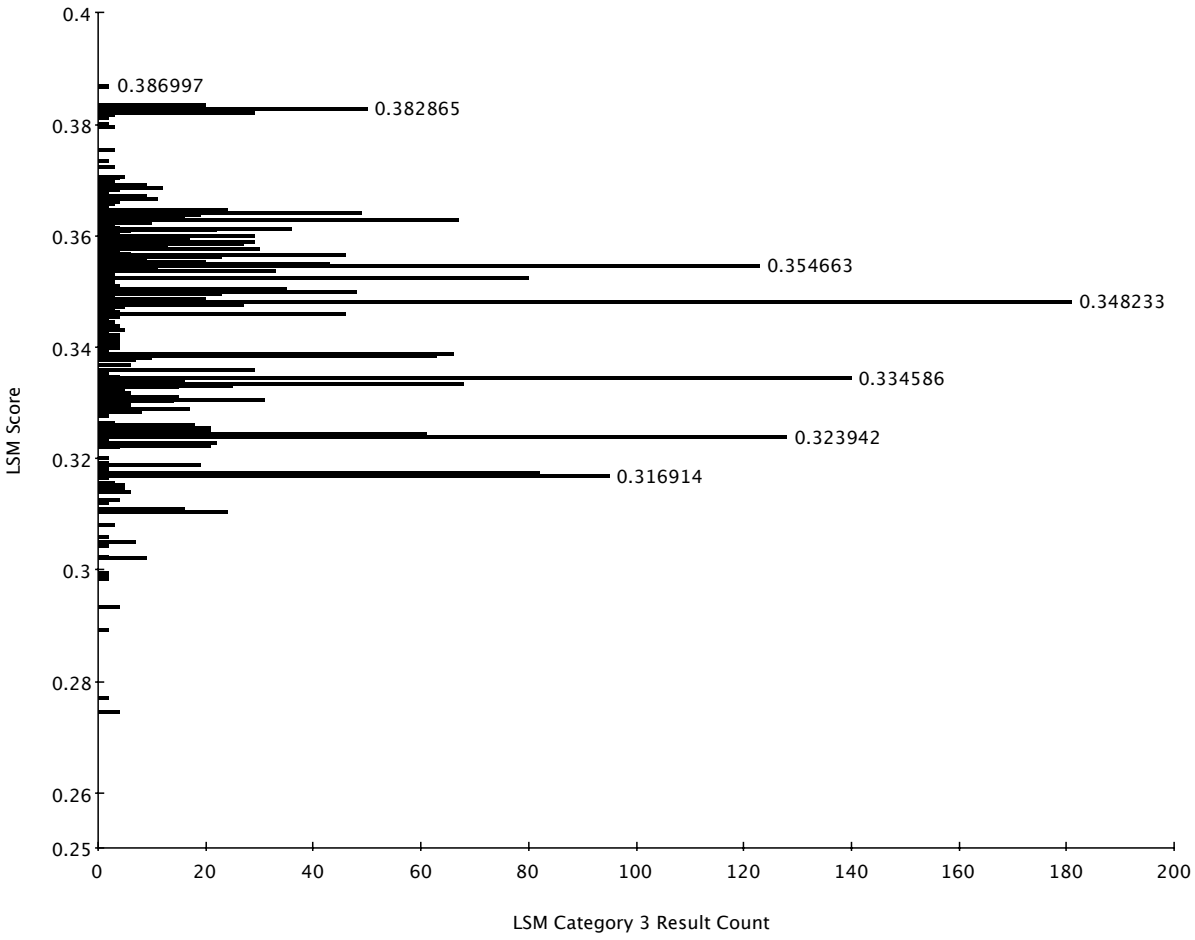

Figure 3.19: Use-Case 2: Selected Results (Category 3)

Table 3.19: Use-Case 2 Category 3 Results

no.	category	score	event
46245	3	0.386997	... Jnpr Syslog 11718 1 [syslog@DOMAIN dayId="20090426" recordId="6" timeRecv="2009/04/26 08:42:53" timeGen="2009/04/26 08:42:53" domain="" devDomVer2="0" ...
46340	3	0.382865	2012–04–29 00:57:47.568185 1177 12238 372 6952 "DOMAIN" 192.168.0.1 200 "Government/Legal;Reference" x_exception_category application/x aspx
47694	3	0.354663	"2012–05–10T16:45:08.000–0700","/var/log/interop/2012/192.168.0.1",syslog,"192.168.0.1","2012–05–10T16:45:08–07:00 192.168.0.1 DCFW 2012–05–10 23:45:13 NOC-LB-BigIP20 F5-LTM ...
48214	3	0.348233	"2012–05–10T16:44:49.000–0700","/var/log/interop/2012/192.168.0.1",syslog,"192.168.0.1","2012–05–10T16:44:49–07:00 192.168.0.1 DCFW 2012–05–10 23:44:54 NOC-LB-BigIP20 F5-LTM ...
49482	3	0.334586	"2012–05–10T16:38:10.000–0700","/var/log/interop/2012/192.168.0.1",syslog,"192.168.0.1","2012–05–10T16:38:10–07:00 192.168.0.1 unbound: [945:1] query: [default] 192.168.0.1:18716 TTL:255 RID:0 DNSSEC:0 None IN PTR DOMAIN. ans: <none> "
50059	3	0.323942	"2012–05–10T16:42:05.000–0700","/var/log/interop/2012/192.168.0.1",syslog,"192.168.0.1","2012–05–10T16:42:05–07:00 192.168.0.1 unbound: [945:0] query: [default] 192.168.0.1:61949 TTL:62 RID:0 DNSSEC:0 None IN A DOMAIN. ans: DOMAIN. 0 IN A 192.168.0.1 "
50479	3	0.316914	"2012–05–10T16:45:13.000–0700","/var/log/interop/2012/192.168.0.1",syslog,"192.168.0.1","2012–05–10T16:45:13–07:00 192.168.0.1 unbound: [945:1] query: [default] 192.168.0.1:46266 TTL:64 RID:0 DNSSEC:0 None IN A DOMAIN. ans: DOMAIN. 1642 IN CNAME DOMAIN. ans: DOMAIN. 21 IN A 192.168.0.1 "

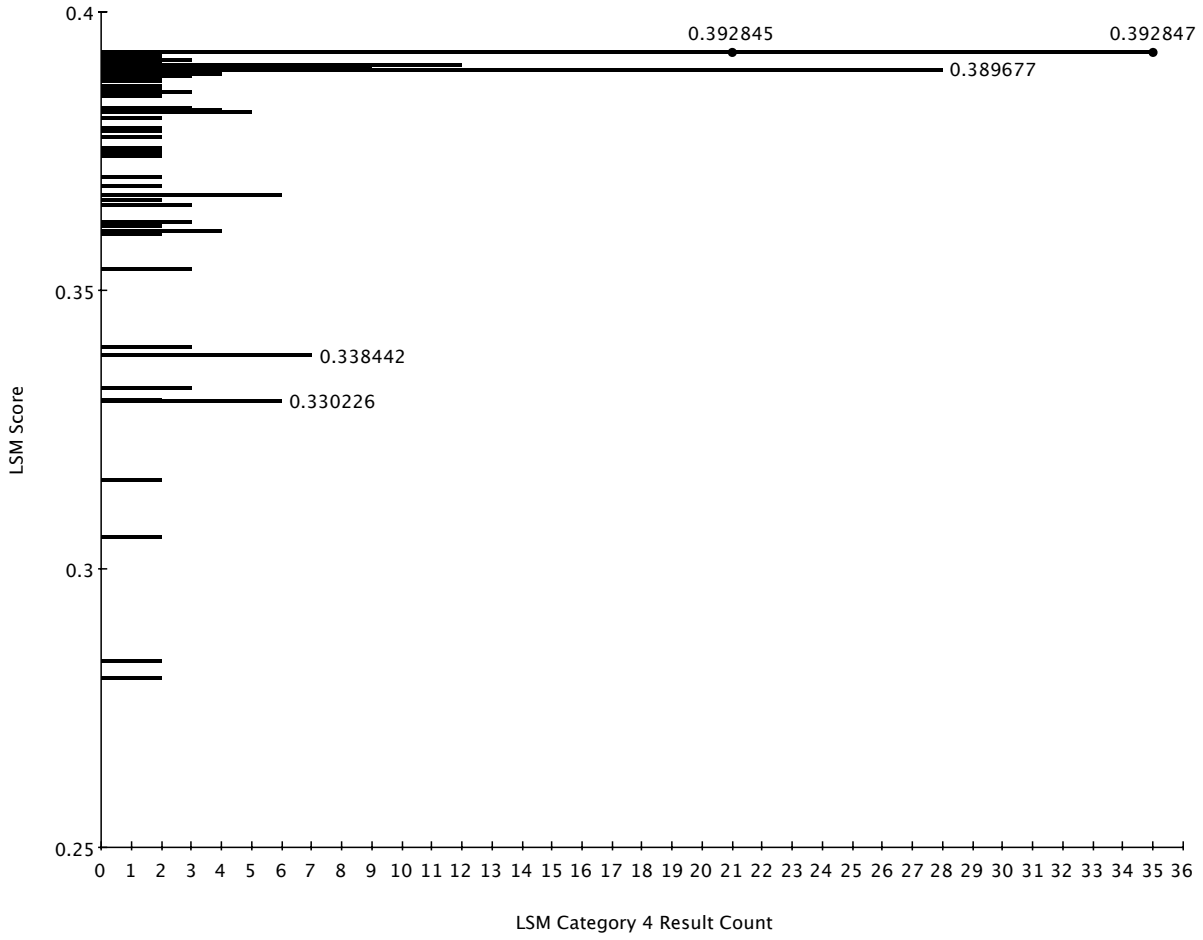

Figure 3.20: Use-Case 2: Selected Results (Category 4)

Category 4 deals with the results from evaluation against Windows Security Table 3.16 events and closely mimics those results of category 2. A sufficient number of true positives allows the evaluation to be classified as a success, and as can be seen in Table 3.20, the false positive events rank well below the true positive results when LSM result scores are compared. The visualisation of the graph seen in Figure 3.20 reflects that with the category 2 analysis, most results are clustered towards the maximum LSM score achieved for this evaluation run.

Table 3.20: Use-Case 2 Category 4 Results

no.	category	score	event
1	4	0.392847	04/25/2012 10:16:10 PM LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4672
35	4	0.392845	04/21/2012 11:47:16 AM LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4672
46119	4	0.389677	04/25/2012 10:30:54 PM LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4634
49320	4	0.338442	** Alert 1333174883.20854: - dovecot, 2012 Mar 31 08:21:23 mx->/var/log/syslog Rule: 9706 (level 3) -> 'Dovecot Session
49843	4	0.330226	Information,2012/05/28 05:34:09 PM,WDSIMGSRV,256,(1),"The description for Event ID 256 from source WDSIMGSRV cannot be found.

3.6.4 Findings

Overall, the results from this experiment show that LSM can certainly play a valuable role when sourcetype classification is needed during SIEM implementation or as part of monitoring continued compliance of a SIEM solution based on its original sourcetype configurations.

One of the most valuable insights gleamed from this experiment is the value of visualisation of the results. Compare Figure 3.17 and Figure 3.19 against the visualisation of Figure 3.19 and Figure 3.20. The graphs clearly assist the user in determining the quality of the LSM results and could be a valuable input during production use to assist in the training of an LSM classifier, or in the refinement of training data. It may be possible to algorithmically determine the quality of category matching in a similar fashion as the visualisations, but this falls outside the scope of this research.

Another improvement that may be considered would be the introduction of a counter-sample category in this use-case, as was found in Use-Case 1 section 3.5. This would imply that misclassified events would then begin to match a category potentially named "Unknown/Counter" and could easily be filtered out to be excluded from other positive results. In an environment where interest is limited to a small selection of sourcetypes,

this may be sufficient.

However, it seems there may be a process whereby a user will continue to iterate on the data and create additional sourcetype categories as false positives are identified through a continuous process of refinement. This would result in the creation of many more categories and over time will lead to most, if not all sourcetypes, being classified correctly. This approach makes more intuitive sense over the long run; instead of classifying events as 'wrong', an event specialist would rather take a little extra time and properly classify the events that were misclassified.

A hybrid approach can also be followed, but as seen, over-enthusiasm in the creation of a counter-sample category would not be recommended at this stage.

3.7 Use-Case 3: Detect Different Sourcetypes and Eventtypes Using Clustering

Security analysts are often faced with historical or archived data, particularly when doing incident response or log forensics. As has been discussed, log volumes can often be overwhelming. Multiple eventtypes and sourcettypes can be found in large collections of thousands or even millions of events. If the data is not classified or normalised, it could be very challenging for an event specialist to determine what events are being dealt with.

In Verizon 2012 Data Breach Investigations Report (Baker *et al.*, 2012, p. 54), the value of looking at larger than normal volumes of logs is underlined. Volume is a great indicator, but a large spike in Remote Authentication related messages might potentially be drowned in an even larger number of other kinds of messages; the ability to do a type-by-type analysis clearly cannot be discounted.

The aim of the experiment is to determine whether or not LSM can be used in event disambiguation without human intervention and whether or not the quality of the clustering is of a sufficient standard to be useful to security practitioners. The advantage of disambiguation of the types of messages is that an analyst can quickly 'get a handle', so to speak, on the types and amount of data in scope for investigation.

Two variations of the experiment were completed to determine the importance of pre-processing when applied to clustering. Ideally, little pre-processing or training would be

needed to produce the desired results; however, based on previous experiments, it was anticipated that the quality of the results would be affected by pre-processing. Subsequently, variation 1 of the experiment did not apply any pre-processing to the text; variation 2 applied the "AdvancedRegexStrategy" as discussed in section 3.5.

3.7.1 Design

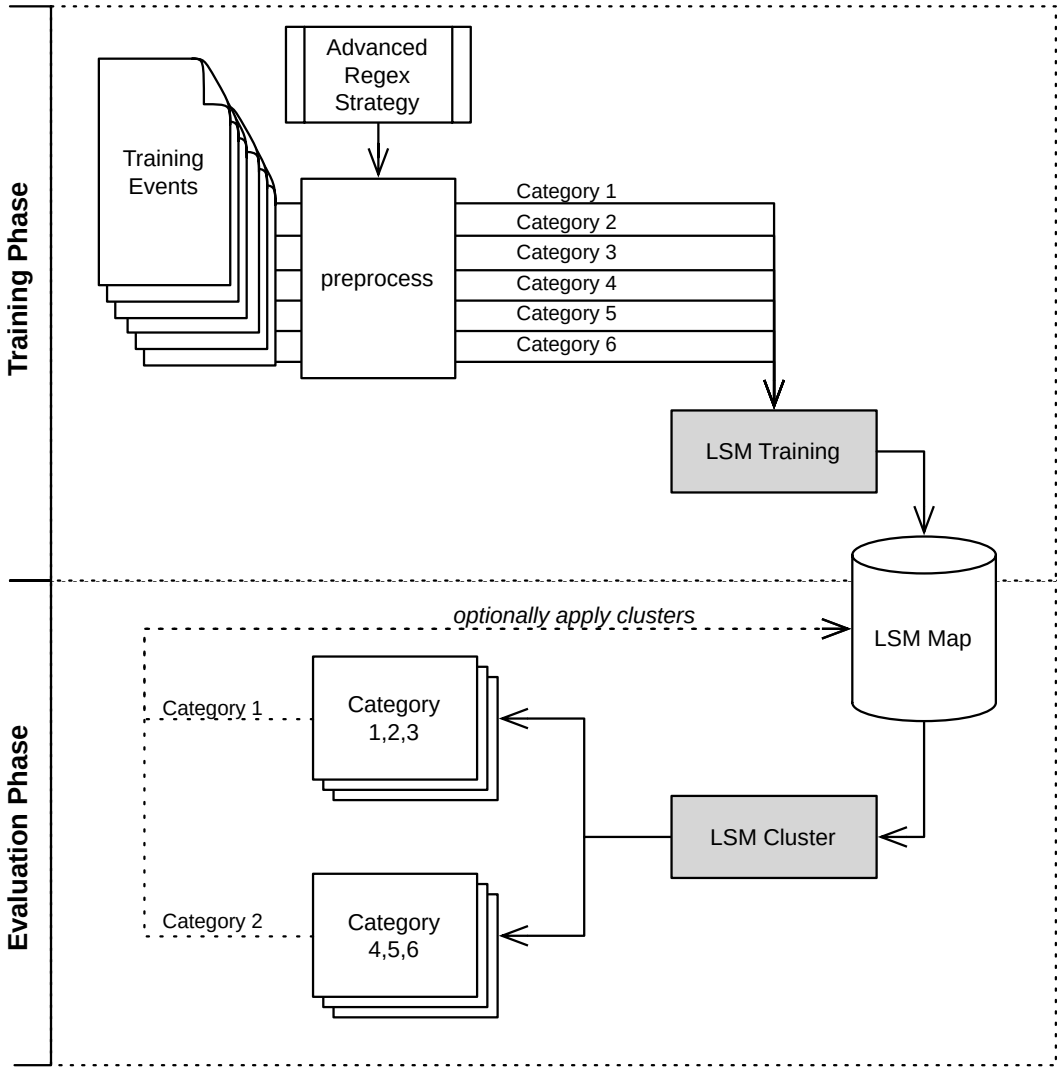

Figure 3.21: Use-Case 3 High Level Process Flowchart

Use-Case 3 re-uses several familiar components used in Use-Case 1 and Use-Case 2, discussed in detail in section 3.5. Pre-processing and training proceeded as normal with the major difference for clustering being that a new LSM category was created for each

event that was trained. In the experiment, 3000 events from the Interop data set were trained and then clustered using the LSM API's agglomerative clustering algorithm (Xu *et al.*, 2003, p. 268). For larger or massive data sets, k-means clustering can be used as it often performs faster than agglomerative clustering. It is possible to re-apply clustering to an LSM map multiple times, as illustrated in the evaluation phase of Use-Case 3 (Figure 3.21).

Recommended practice for LSM is that stop-words and stemming be applied in order to strip a document of any elements that could distract from its latent semantic meaning, as discussed in section 3.3. The first variation of the experiment was set up as deliberately naive: no stemming, stop-words or any modifications were applied. This was meant to serve as a baseline for comparison to the second variation, as a more realistic test was set up within which pre-processing was used.

As with the other experiments, an LSM dimensionality of 200 was selected and bi-grams were generated for the LSM map. The LSM classifier was instructed to reduce the LSM map to no more than 100 categories using clustering, based on the 3000 categories that existed after initial training of the map.

3.7.2 Anticipated Findings

One of the known features of LSM is its ability to disambiguate document types from each other without human intervention. It was anticipated that LSM should perform well with this experiment; however, there was a concern that event disambiguation may not work as well as document disambiguation. Documents often contain hundreds or multiple thousands of words, so clearly, documents relating to economics should be easily distinguishable from documents relating to farming due to co-occurrence of words that are commonly used within those domains. As discussed, though, events are special compositions and as such cannot be treated as normal documents; this may have impacted on the performance of the LSM clustering.

3.7.3 Results

This section contains the results for this Use-Case. It is divided into two subsections in order to address the two variations of the experiment: 1) Variation 1 which did not use pre-processing; and 2) Variation 2 which used the "AdvancedRegexStrategy" pre-processor.

Variation 1

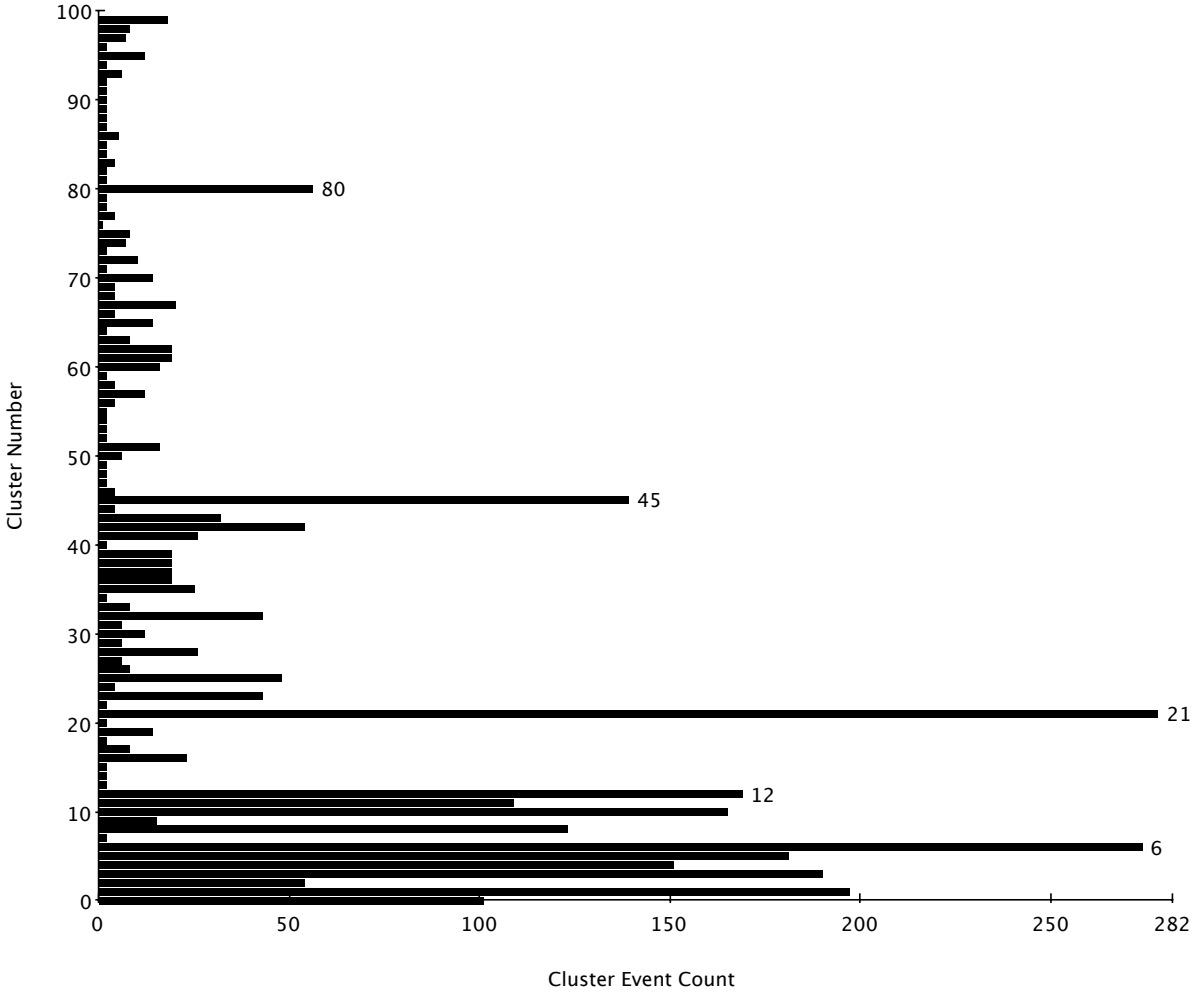

Figure 3.22: LSM Clustering Variation 1 - Raw Data

A visualisation of the event count per cluster can be seen in Figure 3.22. This visualisation was used to more closely investigate the result set and verify the quality of the disambiguation that occurred. Some interesting findings are highlighted in Table 3.21. Most of the clusters created by the LSM classifier contained almost uniform eventtypes as noted in the column 'Dominant Eventtype'; the percentage reflects the number of events of the dominant type, shown in the column named 'Eventtype', that occurred in the cluster.

As shown, multiple numbers of the investigated clusters contained the same eventtype: for example, F5-BigIP events occurring in cluster 6, 21 and 45. It was difficult to determine if the events in these clusters were of the same eventtype, but it was noted that the sourcetypes of the events were the same although they did not all cluster into the same group.

Table 3.21: Clustering Variation 1 Selected Results Summary

Cluster no.	Count	Eventtype	Dominant Eventtype
6	274	F5-BigIP	96%
12	169	Unknown IPS or VPN Device	96%
21	278	F5-BigIP	92%
45	139	F5-BigIP	97%
80	56	Bind DNS	96%
		—	
50	6	crond syscheck	100%
75	8	OpenVPN TLS Error	100%
98	8	OpenVPN	100%

Of interest is that cluster 80, Bind DNS, also contained Unbound DNS events, an alternative DNS server. This was due to the similar underlying semantic structure for DNS events. It can therefore be stated that cluster 80 contained multiple sourcetypes but a single eventtype.

Some smaller clusters were also chosen and summarised; as can be seen, some of the smaller clusters contained 100% of the same eventtype.

Overall, the quality of the results was encouraging and even though there were multiple clusters that contained the same sourcetype, as with the F5-BigIP clusters, the results were still easily consumable.

Variation 2

Table 3.22: Clustering Variation 2 Selected Results Summary

Cluster no.	Count	Eventtype	Dominant Eventtype
4	570	Unbound DNS	98%
17	284	F5-BigIP IPv4	98%
24	143	F5-BigIP IPv6	98%
70	52	Bind DNS	100%
		—	
0	7	Bind DNS Zone Transfer Request	100%
1	7	Bind DNS Zone Transfer Deny	100%
44	26	Quagga Routing Engine	100%

As with the first variation of this experiment, the a visualisation of the results could be seen in Figure 3.23. This visualisation was used to identify the largest clusters, summarised in Table 3.22 along with other clusters of interest.

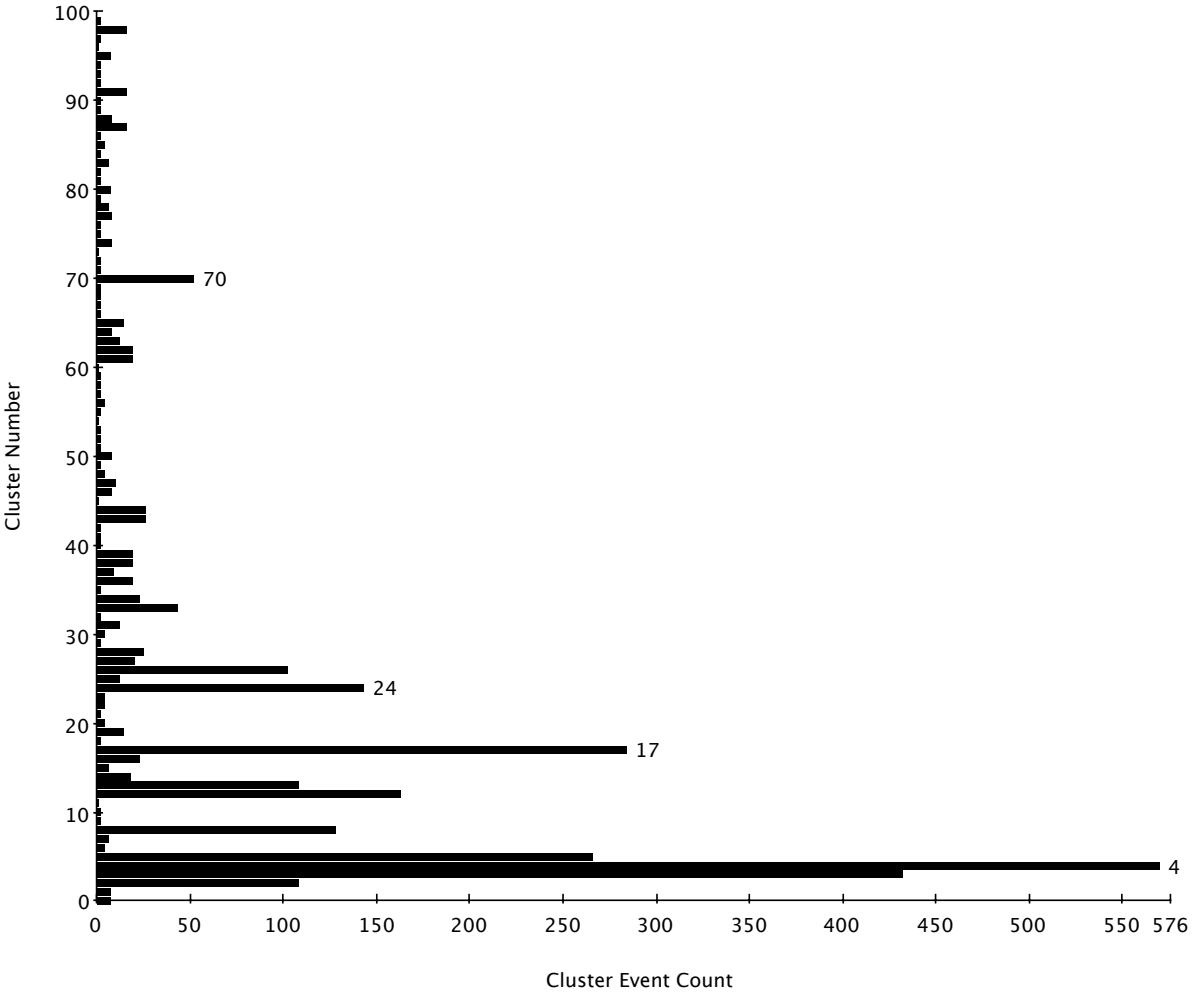

Figure 3.23: LSM Clustering Variation 2 - Pre-processed Data

Table 3.23: Clustering Variation 2 Selected Result Examples

Cluster	Event No.	Raw Event
0	1	2012–05–10T16:45:31–07:00 192.168.0.1 named[752]: client 192.168.0.1#59187: received notify for zone 'DOMAIN'
1	2	2012–05–10T16:45:31–07:00 192.168.0.1 named[752]: zone DOMAIN/IN: refused notify from non-master: 192.168.0.1#59187
17	2980	2012–05–10T16:43:19–07:00 192.168.0.1 DCFW 2012–05–10 23:43:24 NOC-LB-BigIP20 F5-LTM rule-dg-dcf-net-shownetworks 1 allow 17 192.168.0.1 10940 192.168.0.1 162 /Common/DCF-IPv4–192.168.0.1-udp–162 non
24	124	2012–05–10T16:45:24–07:00 192.168.0.1 DCFW 2012–05–10 23:45:29 NOC-LB-BigIP20 F5-LTM rule-dg-dcf-net-any 1 allow 17 2620:144:2d00:200::66 58964 2620:144:2d00:800::174 53 /Common/DCF-IPv6–2620.144.2d0

Overall, the results of the clustering were extremely satisfying: where variation 1 had usable results, variation 2 had clearly partitioned eventtypes for most of the clusters. The co-occurrence of eventtypes in a single cluster was relatively low and did not pose any problems for the analysis. Interesting finds such as the clear difference between two Bind DNS eventtypes in clusters 0 and 1 Table 3.23 highlight the value for any individual tasked with analysing or normalising the data. Another interesting example of the benefit of clustering is apparent with clusters 17 and 24: where with variation 1 there were multiple large clusters of F5-BigIP events, in variation 2 there are two dominant clusters. Closer inspection showed that cluster 17 contains only IP version 4 events and that cluster 24 contains only IP version 6 events. This type of insight into a data set is invaluable when creating parsers and plug-ins for SIEM correlation engines.

Variation 2 has a much cleaner distribution of events, as can be witnessed by inspecting Figure 3.23, the graphical representation of its cluster counts. In this variation, pre-processing was applied and the results indicated that as with the other experiments, pre-processing is an invaluable part of the overall process as it assists noticeably in uncovering the underlying, or latent, semantic structure of events.

3.7.4 Findings

Overall, the results for Use-Case 3 were very encouraging. The benefits of clustering in identifying eventtypes can be seen. Over and above the potential insights that an event specialist could gleam from a large collection of data through the use of clustering, clustering can also assist in identifying eventtypes that could in turn be used for training data in other use-cases.

The clustering performance was very good and the results easily parsed by the user, a great benefit of LSM in the information security domain.

3.8 Summary

This chapter described the approach that was followed in the designing the experiments for our use-cases and how LSM was applied to the area of interest. The data collected for the experiments was then discussed in greater detail. section 3.3 delved into the details of the evolution of the software and processing methodologies that were used to conduct the research. A discussion related to the design of the final software was provided, as well as insights that were garnered relating to data pre-processing. The three primary use-cases and findings related to their experiments were discussed in section 3.5, section 3.6 and section 3.7 respectively. In chapter 4, the results from the experiments as well as the methodology used will be considered.

Chapter 4

Evaluation of Results

This chapter deals with the outcome of the experiments and considers the metrics and methodology used during the evaluation of the results. In section 4.1, the particular metrics used to evaluate the results are considered. Important insights gained regarding the processing architecture and the software available for this research are both examined in section 4.2. The chapter concludes with section 4.3 wherein the successes achieved during experimentation are summarised.

4.1 Evaluation Metrics

For the purposes of result analysis and evaluation, metrics had to be selected and interpreted. LSM produces a set of results for each evaluation that is completed against an LSM map, results which are a sorted list of LSM scores and their associated category. The category with the largest score is listed first and can be interpreted as the 'most likely category match' for the given evaluation data. The top result based on highest score was used for all evaluation.

Use-Cases 1 and 2 applied this metric and dealt with the likelihood of an event belonging to a set of pre-trained categories; as such, it was deemed that the top score would be a sufficient measure. The well-established practices of calculating recall, precision and the f-measure of the results allowed further confirmation of the results to be obtained.

Another important measurement was a 'per-category' or 'per-cluster' count as with Use-Case 3. The number of results within a category or cluster were visualised in a set of graphs

that provided valuable insight into the composition of the clusters and the relevancy and accuracy of the results. In Use-Case 2, visualisation displayed clearly to the user whether or not the results for the category match were highly concentrated towards the top LSM scores or rather more distributed and thus less accurate.

Visualisation was used to aid in the interpretation and selection of results for Use-Cases 2 and 3 and as such, a detailed analysis of the interpretation of the LSM result scores was not conducted, although as discussed in Use-Case 2, this may warrant further investigation for future research.

4.2 Evaluation of Experimental Procedures

The areas of interest for this research – log management and SIEM – more often than not deal with very large volumes of data. When these are taken into account, the challenges faced with data and result management, challenges which prompted several redesigns of the software used in the research, had to be acknowledged. These experiments relied heavily on sampling and focussed on proof of concepts rather than an evaluation at production scale.

There are other technologies in this space such as Bayesian spam filters (Androutsopoulos *et al.*, 2000) that have been demonstrated to work at scale; for example, as used with spam filters on large email gateways, LSM was not evaluated under those circumstances in this research. The scale at which the research was conducted does not make it incomparable to production use; rather, most real world implementation and investigation dealing with events may easily be conducted on the same scale as this research. The lessons learnt and insights awarded from such sampling and investigation may then be used with other technologies at scale.

Manual searching and the use of other technologies such as regular expressions may also lead to similar results; it is the ease of use of LSM once a software framework is in place that makes it very attractive for real world use. The relative forgiving nature of LSM when it comes to training data and sample selection once the correct pre-processing is applied makes LSM very attractive as an alternative technology.

The largest stumbling block at this point is that the LSM software and framework is *only* available on the MacOS platform. Future use in production environments would benefit greatly from a cross platform implementation of LSM.

The LSMPP software took around five minutes to evaluate the 51,000 events in evaluation data in Use-Case 2 section 3.6. An analysis of the performance of the software indicated that a large amount of CPU time was consumed by pre-processing. Furthermore, the software did not benefit from multi-processing that was available on the platform. Fifty one thousand events are a relatively small number of events in the context of a production log management environment and would, under the current performance characteristics, relegate the use of the tool to smaller or sample sets of data.

4.3 Use-Case Successes

When viewed narrowly as designed, Use-Case 1 (section 3.5) did not achieve great success. However, after requirements were relaxed and the scope of the evaluation broadened from "logon success" events to "logon" events, the results improved dramatically. Training data selection remains a challenge: it is hard to imagine an off-the-shelf classifier being made available at the current implementation levels. Use-Case 1 illustrates the potential value for event identification but does not provide a turnkey solution for such a requirement. Use-Case 2 (section 3.6) also highlighted the relative sensitivity of the technology to training data selection. The addition of visualisation made the interpretation of results tractable and valuable insights into the sourcetypes of the data were then able to be obtained. Clustering using LSM displayed great potential for event disambiguation in Use-Case 3 (section 3.7). Pre-processed events clustered very well and with a couple of user interface refinements, LSM could be very useful in this scenario. The required human intervention was minimal and the quality of results was high.

However, LSM does not appear to be accurate enough for event-based processing in the absence of pre-processing and applied domain knowledge. It seems infeasible that a security practitioner would just export some events, run it through an off-the-shelf LSM process and obtain actionable results. LSM combined with sufficient domain knowledge-based pre-processing and careful training data selection used as part of an integrated solution seems to hold a great deal of promise for event processing.

Training data selection can be facilitated with the correct graphical and user interfaces, but yet remains a challenge.

4.4 Summary

This chapter considered the application of LSM to the field of Security Event Analysis, both for assisting security analysts in conducting forensic and other types of log analysis, but also from the point of view of SIEM implementers and how LSM can assist in the challenges faced in that discipline. The discussion covered the results in the context of the pre-processing, insights gained, and the acknowledged importance of careful selection of training data.

The research concludes in the subsequent chapter which revisits the research goals embarked upon in the chapter 1, summarises the finding of the research conducted in chapter 3 and points the way to potential future research opportunities.

Chapter 5

Conclusions

5.1 Significance of Research

In this text, we introduced the challenges faced by event specialists and stated the intent to evaluate LSM in this context in chapter 1.

A comprehensive look into events, their nature and utility, as well as their role in larger log management practice and SIEM systems was then discussed, along with LSM in chapter 2.

This work then elucidated the approach taken by the researcher, challenges faced with data collection and software development, insights obtained from the experiments, as well as the details of the experiments themselves in chapter 3.

Finally, the results as discussed in the Use-Case sections were revisited and summarised in chapter 4.

This chapter concludes the work, discussing the significance of the research and presenting possible future research and development that could be undertaken as a continuance of this work.

The goal of this research was to determine if Latent Semantic Mapping (LSM) could be a useful technology to event specialists when facing the challenges related to large log management and SIEM practices, particularly when dealing with implementation challenges, incident response and forensics. As it stands, these challenges rely heavily on

the ability of the professional to reliably and quickly identify, classify and disambiguate relevant event data.

Several experiments, grouped into three use-cases, were conducted: Use-Case 1 (section 3.5) addressed the challenge of identification of events (LSM performed well after the original requirements were somewhat relaxed and achieved a high level of precision and recall); Use-Case 2 (section 3.6) addressed classification (results from experiments highlighted the value of LSM and established some visual aids that could assist event specialists in processing the relevant results); and finally, Use-Case 3 (section 3.7), explored LSM with appropriate pre-processing and found that it very successfully addressed the challenge of disambiguation.

Overall, it was found that LSM performed well when the appropriate pre-processing and training data selection took place. It also became apparent that LSM would perform best as part of larger collection of tools and strategies available to security practitioners and that it was unlikely to be of sufficient value in the absence of pre-processing and a larger system that facilitated the processing of events and results. The unintegrated LSM CLI tools would be of little value to security practitioners.

Table 5.1: Table of Core Findings

Use-Case	Test	Deals With	Findings
1	Detect A Trained Eventtype	Identification	Successful. Depends heavily on training data. F-measure of more than 0.91 achieved for three of the five training sets.
2	Detect Multiple Sourcetypes From A Single Stream	Classification	With the aid of visualisation provides valuable insights to the classified data, aids significantly in manual classification.
3	Detect Different Sourcetypes and Eventtypes Using Clustering	Disambiguation	Very successful with the correct preprocessing.

It is clear from the research conducted that Latent Semantic Mapping is an exciting paradigm which most definitely has a role to play in the Information Security field. Once understood and integrated into a sufficient software architecture, very challenging workflows could be automated as well as insights achieved on event data that would have been difficult to achieve through traditional means and may have been impractical to achieve when faced with work and time pressure.

An important insight gleaned from the development of the software and processing architecture was that events cannot be treated as normal documents (section 3.3) even though events are unstructured data; with event data punctuation and other language artefacts that may detract from the underlying latent semantic meaning of a normal document actually becomes very valuable and forms an important part of the structure and the latent semantic meaning of an event.

Due to this special nature of events, the pre-processing model was designed to capture and amplify the structure of events. This led to the development of specialised pre-processing routines and was a critical success factor for other LSM-related work in this domain.

LSM would also benefit greatly from being applied as a part of a larger workflow of event classification, identification and disambiguation. There is little practical reason to use LSM on well-known and well-understood event data that has already been identified, classified and disambiguated. Consequently, a user could scale down the need for LSM and eliminate a great deal of noise that may impact the performance of the LSM classifier.

This leads to the conclusion that LSM use in this space would greatly benefit from interactivity that would easily allow for multiple iteration runs of the processes as discussed in this research. The ability to tag, classify, refine or tweak and then re-run would require a specialised software environment.

At this stage, it is not feasible to envision that LSM would be integrated directly into large scale log management system or SIEMs, but rather that it would be integrated into reporting and management consoles of such systems.

Interactivity and the ability to re-run and iterate through results would assist a user in selecting the correct training data for production. The importance of good training data was well-understood from other LSM uses and was re-established as critically important in this research, particularly in the context of Use-Case 2 and Use-Case 3. Combined with visualisation, as in Use-Case 2, the solution may prove to be more flexible and user-friendly than other systems that require in-depth knowledge of regular expression syntax.

Another challenge with the use of LSM is obtaining sufficient training data and samples. This might be offset in a production system by the availability of such samples and data in production log management and in SIEM systems; it is also likely that the identification, categorisation and disambiguation that is needed in production would relate to data that is already available in the organisation. An off-the-shelf solution would have to contain trained maps based on ample training data at the originator in a scenario where LSM is integrated into a larger software based solution.

In summary, as part of an interactive, iterative process of Security Event Analysis, LSM holds great promise for provision of valuable insights and understanding of diverse collections of event information specifically to event specialists. The research has shown this clearly from the experiments conducted.

5.2 Future Work

This final section considers possible further research and developments based on this research and the application of LSM.

- Further research is warranted into the ideal pre-processing processes needed to use LSM with event data. Some inroads have been made into this topic as a part of this research, but it is by no means comprehensive. Pre-processing has a powerful impact on the quality of the results and an in-depth examination of the algorithmic interaction between LSM and data formats could lead to more powerful insights and vastly improved results. Experimentation with different dimensionality settings, lexicalisation and the use of tri-grams or other constructs could lead to further refinement of the results, particularly now that this research provides the evidence that further, detailed investigation is warranted.

- The research also deals with a limited number of use-cases. Based on the findings in this research, an extensive evaluation of the existing work with multiple use-cases could provide powerful new directions for research and lead to further insights into the special nature of machine data and its properties.

- As shown in Use-Cases 2 and 3, visualisation of the results provides valuable insights into the quality of the results. This work could be further extended and form the basis of an algorithmic solution to detect the quality of results, leading to the development of intelligent feedback systems that could assist in training data fine-tuning and selection.

- This body of work also did not evaluate the applicability of LSM at scale. Research into this area could evaluate the performance characteristics of LSM and determine if it has a future as an integrated part of a log management system or SIEM.

- During the data collection phase, it became apparent that a public source of anonymised event and log samples does not appear to exist at present. Such a solution would

be invaluable to enable further research into this area, as it would remove a major impediment that future researchers would face and also assure richness in test data and test data coverage.

- Research into the applicability of other natural language and artificial intelligence paradigms to the research and use-cases in this work could also be undertaken.

The world is steadily embracing more and more data, and with this deluge – this Big Data – data processing has become substantially more challenging. Machine data in the form of events make up a large percentage of the data that needs to be processed and stored on a daily basis. This field is still ripe for research, innovation and development.

References

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C. D., and Stamatopoulos, P. *Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. arXiv preprint cs/0009009*, 2000. [Online; last accessed 13-December-2012].
URL `http://arxiv.org/abs/cs/0009009`

Bagwill, R., Carnahan, L., Kuhn, R., Nakassis, A., Ransom, M., Backley, J., Chang, S.-j., Markovitz, P., Olsen, K., and Wack, J. *NIST Special Publication 800-7*. 1995. doi:10.1.1.3.7701. [Online; last accessed 30-November-2012].

Baker, W., Goudie, M., Hutton, A., Hylender, C. D., Niemantsverdriet, J., Novak, C., Ostertag, D., Porter, C., Rosen, M., and Tippett, P. *2012 DATA BREACH INVESTIGATIONS REPORT*. 2012. [Online; last accessed 30-November-2012].
URL `http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf`

Bejtlich, R. *TaoSecurity: Defensible Network Architecture 2.0*. 2008. [Online; last accessed 30-November-2012].
URL `http://taosecurity.blogspot.com/2008/01/defensible-network-architecture-20.html`

Bellegarda, J. *A multispan language modeling framework for large vocabulary speech recognition. Speech and Audio Processing, IEEE Transactions on*, 6(5):456–467, 1998. doi:10.1109/89.709671.

Bellegarda, J. *Latent semantic mapping: dimensionality reduction via globally optimal continuous parameter modeling.* In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 127 –132. nov. 2005a. doi:10.1109/ASRU.2005.1566490.

Bellegarda, J., Butzberger, J. W., Chow, Y.-L., Coccaro, N. B., and Naik, D. *A novel word clustering algorithm based on latent semantic analysis.* In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, pages 172–175. 1996. doi:10.1109/ICASSP.1996.540318.

Bellegarda, J., Naik, D., and Silverman, K. E. A. *Automatic junk e-mail filtering based on latent content. Automatic Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop on*, pages 465–470, 2003. doi:10.1109/ASRU.2003.1318485.

Bellegarda, J. and Silverman, K. E. A. *Natural language spoken interface control using data-driven semantic inference. Speech and Audio Processing, IEEE Transactions on*, 11(3):267–277, 2003. doi:10.1109/TSA.2003.811534.

Bellegarda, J. R. *Latent semantic mapping [information retrieval]. Signal Processing Magazine, IEEE*, 22(5):70–80, 2005b. doi:10.1109/MSP.2005.1511825.

Bellegarda, J. R. *Latent Semantic Mapping: Principles and Applications (Synthesis Lectures on Speech and Audio Processing)*. Morgan and Claypool Publishers, March 2008. ISBN 1598291041.

Benton, C., Bird, T., and Ranum, M. J. *Top 5 Essential Log Reports.* 2006. [Online; last accessed 30-November-2012].
URL `http://www.sans.org/security-resources/top5_logreports.pdf`

Berry, M., Dumais, S., and O'Brien, G. *Using linear algebra for intelligent information retrieval. SIAM Review*, 37(4):573–595, 1995. doi:10.1137/1037127.

Binde, B. E., McRee, R., and O'Connor, T. J. *Assessing Outbound Traffic to Uncover Advanced Persistent Threat.* 2011. [Online; last accessed 30-November-2012].
URL `http://www.sans.edu/student-files/projects/JWP-Binde-McRee-OConnor.pdf`

Bitincka, L., Ganapathi, A., Sorkin, S., and Zhang, S. *Optimizing data analysis with a semi-structured time series database. SLAML '10: Proceedings of the 2010 workshop on Managing Systems Via Log Analysis and Machine Learning techniques*, 2010.

Borthakur, D., Gray, J., Sarma, J. S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., Schmidt, R., and Aiyer, A. *Apache hadoop goes realtime at facebook.* In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11,

pages 1071–1080. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0661-4. doi: 10.1145/1989323.1989438.

Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., and Robinson, W. *Performance Measurement Guide for Information Security SP800-55. National Institute of Standards and Technology (USA).* 2008. [Online; last accessed 30-November-2012].
URL `http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf`

Chuvakin, D. A. *Updated With Community Feedback SANS Top 7 Essential Log Reports DRAFT2.* 2010. [Online; last accessed 30-November-2012].
URL `http://chuvakin.blogspot.com/2010/08/updated-with-community-feedback-sans_06.html`

Chuvakin, D. A., Borchardt, P., Buley, M., Melson, P., and Kohli, V. *Navigating the Data Stream Without Boiling the Ocean.* 2010. [Online; last accessed 30-November-2012].
URL `http://newserver.iansresearch.com/research/infrastructure-security/navigating-data-stream-without-boiling-ocean-case-studies-effective`

Cichonski, P., Millar, T., Grance, T., and Scarfone, K. *Computer Security Incident Handling Guide SP800-61 (Draft). National Institute of Standards and Technology (USA).* January 2012. [Online; last accessed 30-November-2012].
URL `http://csrc.nist.gov/publications/drafts/800-61-rev2/draft-sp800-61rev2.pdf`

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. *Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science*, 41(6):391–407, 1990.
URL `http://www.cob.unt.edu/itds/faculty/evangelopoulos/dsci5910/LSA_Deerwester1990.pdf`

Federal Communications Commission. *FCC Computer Security Incident Response Guide.* July 2002.
URL `http://csrc.nist.gov/groups/SMA/fasp/documents/incident_response/Incident-Response-Guide.pdf`

Frei, S., May, M., Fiedler, U., and Plattner, B. *Large-scale vulnerability analysis.* In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, LSAD

'06, pages 131–138. ACM, New York, NY, USA, 2006. ISBN 1-59593-571-1. doi: 10.1145/1162666.1162671.

Fry, A. *A Forensic Web Log Analysis Tool: Techniques and Implementation.* Master of Information Systems Security, Concordia University, 2011.
URL `http://spectrum.library.concordia.ca/7769/1/Fry_MASc_F2011.pdf`

Gong, Y. and Liu, X. *Generic text summarization using relevance measure and latent semantic analysis.* In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 19–25. ACM, New York, NY, USA, 2001. ISBN 1-58113-331-6. doi:10.1145/383952.383955.

Gotoh, Y. and Renals, S. *Document space models using latent semantic analysis.* In *Proceedings Eurospeech, pages 1443-1446, Rhodes, 1997.* 1997.
URL `http://lac-repo-live7.is.ed.ac.uk/handle/1842/1222`

Goyvaerts, J. and Levithan, S. *Regular Expressions Cookbook.* O'Reilly Media, Sebastopol, CA, USA, May 2009. ISBN 9780596520687.

Hafner, K. *Researchers Yearn to Use AOL Logs, but They Hesitate - New York Times.* August 2006.
URL `http://www.nytimes.com/2006/08/23/technology/23search.html?ex=1313985600&en=cc878412ed34dad0&ei=5088&partner=rssnyt&emc=rss`

Harris, S. *CISSP All-in-One Exam Guide, 6th Edition.* McGraw-Hill Osborne Media, New York, NY, USA, February 2012. ISBN 9780071781732.

Hayden, L. *IT security metrics: A practical framework for measuring security & protecting data.* McGraw-Hill Osborne Media, New York, NY, USA, 2010. ISBN 0071713409.

Hull, R. D., Singh, S. B., Nachbar, R. B., Sheridan, R. P., Kearsley, S. K., and Fluder, E. M. *Latent Semantic Structure Indexing (LaSSI) for Defining Chemical Similarity. Journal of Medicinal Chemistry*, 44(8):1177–1184, April 2001. doi:10.1021/jm000393c.

Iwata, T., Yamada, T., and Ueda, N. *Probabilistic latent semantic visualization: topic model for visualizing documents.* In *KDD '08: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 363–371. 2008. ISBN 1605581933. doi:10.1145/1401890.1401937.

Jacob, A. *Big Data @ Disney*. 2012. [Online; last accessed 30-November-2012].
URL http://www.datastax.com/wp-content/uploads/2012/08/C2012-BigDataatDisney-ArunJacob.pdf

Jolliffe, I. T. *Principal Component Analysis 2nd edition*. Springer, 2002. ISBN 0387954422.

Josephes, C. *Writing Apache's Logs to MySQL - O'Reilly Media*. 2005. [Online; last accessed 30-November-2012].
URL http://onlamp.com/pub/a/apache/2005/02/10/database_logs.html

Karlzén, H. *An Analysis of Security Information and Event Management Systems*. Master's Thesis, Chalmers University of Technology, 2008.
URL http://spectrum.library.concordia.ca/7769/1/Fry_MASc_F2011.pdf

Kazakow, P. *An Investigation of Latent Semantic Mapping of Ontologies*. Diplomarbeit, TU Hamburg-Harburg, June 2008.
URL http://www.sts.tu-harburg.de/pw-and-m-theses/2008/kaza08.pdf

Kent, K. and Souppaya, M. P. *Guide to Computer Security Log Management SP800-92. National Institute of Standards and Technology (USA)*. September 2006. [Online; last accessed 01-December-2012].
URL http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf

Kim, Y.-S., Chang, J.-H., and Zhang, B.-T. *An empirical study on dimensionality optimization in text mining for linguistic knowledge acquisition*. In *Proceedings of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'03, pages 111–116. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 3-540-04760-3.
URL http://dl.acm.org/citation.cfm?id=1760894.1760910

Koivunen, E. *Effective information sharing for incident response coordination*. Master of science in technology, Aalto University, 2010.
URL http://personal.inet.fi/koti/erka/Studies/DI/DI_Erka_Koivunen.pdf

Landauer, T. and Dumais, S. *A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological Review; Psychological Review*, 104(2):211, 1997.

Landauer, T. K., Foltz, P. W., and Laham, D. *An introduction to latent semantic analysis. Discourse Processes*, 25(2-3):259–284, 1998. doi:10.1080/01638539809545028.

Lobo, P. V. and de Matos, D. M. *Fairy tale corpus organization using latent semantic mapping and an item-to-item top-n recommendation algorithm.* In *LREC'10: Proceedings of the seventh International Conference on Language Resources and Evaluation*, pages 1472 – 1475. 2010.
URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/786_Paper.pdf

Lämmel, R. *Google's mapreduce programming model — revisited. Science of Computer Programming*, 70(1):1 – 30, 2008. ISSN 0167-6423. doi:10.1016/j.scico.2007.07.001.

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, 2008. ISBN 9780521865715.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. *Big data: The next frontier for innovation, competition, and productivity.* 2011. [Online; last accessed 01-December-2012].
URL http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation

Marty, R. *Security Intelligence and Big Data. raffy.ch*, 2007. [Online; last accessed 2-December-2012].
URL http://raffy.ch/blog/2007/08/25/event-processing-normalization/

Marty, R. *Cloud application logging for forensics.* In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 178–184. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0113-8. doi:10.1145/1982185.1982226.

Microsoft. *Security Event ID: 540.* 2012. [Online; last accessed 13-December-2012].
URL http://www.microsoft.com/technet/support/ee/transform.aspx?ProdName=Windows+Operating+System&ProdVer=5.0&EvtID=540&EvtSrc=Security&LCID=1033

Nair, S. *The art of database monitoring. ISACA*, 3:1 – 4, 2008. [Online; last accessed 01-December-2012].
URL http://www.isaca.org/Journal/Past-Issues/2008/Volume-3/Documents/jopdf0803-art-of-database.pdf

Payne, S. C. *A Guide to Security Metrics.* 2006. [Online; last accessed 03-December-2012].
URL http://www.sans.org/reading_room/whitepapers/auditing/guide-security-metrics_55

Puffinware. *Latent Semantic Analysis (LSA) Tutorial*. January 2010. [Online; last accessed 03-December-2012].
URL `http://www.puffinwarellc.com/index.php/news-and-articles/articles/33.html`

Riloff, E. and Lehnert, W. *Information extraction as a basis for high-precision text classification. Transactions on Information Systems (TOIS)*, 12(3), July 1994. doi: 10.1145/183422.183428.

Rowlingson, R. *A ten step process for forensic readiness. International Journal of Digital Evidence*, 2:1 – 28, 2004.
URL `http://www.utica.edu/academic/institutes/ecii/publications/articles/A0B13342-B4E0-1F6A-156F501C49CF5F51.pdf`

Salton, G., Wong, A., and Yang, C. S. *A vector space model for automatic indexing. Commun. ACM*, 18(11):613–620, November 1975. ISSN 0001-0782. doi:10.1145/361219.361220.

Secmon. *ArcSight Flex Connector Methodology*. 2012. [Online; last accessed 03-December-2012].
URL `http://edgeseven.com/downloads/arcsight-flex-connectors.pdf`

Shen, X. *Chronicle of AOL Search Query Log Release Incident*. pages 1–2, 2012. [Online; last accessed 03-December-2012].
URL `http://sifaka.cs.uiuc.edu/xshen/aol_querylog.html`

Shenk, J. *SANS Sixth Annual Log Management Survey Report*. 2010. [Online; last accessed 03-December-2012].
URL `http://www.sans.org/reading_room/analysts_program/rsa-management-survey-june-2010.pdf`

Silva, C., Cignoli, C., and Friedman, J. *IANS Trends: A Behavioral Approach To Threat Modeling*. June 2011. [Online; last accessed 03-November-2012].
URL `http://www.iansresearch.com/research/security-operations/vulnerability-threat-management/ians-trends-behavioral-approach-threat-`

SMNP. *Internet Engineering Task Force RFC 3418*. 2012. [Online; last accessed 13-December-2012].
URL `http://tools.ietf.org/html/rfc3418`

Sorkin, S. *Large-Scale, Unstructured Data Retrieval and Analysis Using Splunk.* Technical report, Splunk Inc., 2009. [Online; last accessed 03-December-2012].
URL `http://www.splunk.com/web_assets/pdfs/secure/Splunk_and_MapReduce.pdf`

Splunk. *Overview of search-time field extraction.* 2012. [Online; last accessed 03-December-2012].
URL `http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Addfieldsatsearchtime`

Stearley, J., Corwell, S., and Lord, K. *Bridging the gaps: joining information sources with splunk.* In *Proceedings of the 2010 workshop on Managing Systems via Log Analysis and Machine Learning techniques*, SLAML'10, pages 8–8. USENIX Association, Berkeley, CA, USA, 2010.

Swift, D. *A Practical Application of SIM/SEM/SIEM Automating Threat Identification.* 2006. [Online; last accessed 03-December-2012].
URL `http://www.sans.org/reading_room/whitepapers/logging/practical-application-sim-sem-siem-automating-threat-identification_1781`

Syslog. *Internet Engineering Task Force RFC 5424.* 2012. [Online; last accessed 13-December-2012].
URL `http://tools.ietf.org/html/rfc5424`

Tang, C., Xu, Z., and Dwarkadas, S. *Peer-to-peer information retrieval using self-organizing semantic overlay networks.* In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 175–186. 2003. ISBN 1581137354. doi:10.1145/863955.863976.

Tarala, J. *Implementing the 20 Critical Controls with Security Information and Event Management (SIEM) Systems.* April 2011. [Online; last accessed 03-December-2012].
URL `http://www.sans.org/reading_room/analysts_program/siem-systems-arcsight.pdf`

Uyar, A. *Google stemming mechanisms. Journal of Information Science*, 35(5):499–514, 2009. doi:10.1177/1363459309336801.

Van de Cruys, T. and Apidianaki, M. *Latent semantic word sense induction and disambiguation.* In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages

1476–1485. Association for Computational Linguistics, Stroudsburg, PA, USA, 2011. ISBN 978-1-932432-87-9.

Wicijowski, J. and Ziółko, B. *Extracting Semantic Knowledge from Wikipedia.* In *IIS'2010: Proceedings of Intelligent Information Systems 2010*, pages 91 – 98. 2010. ISBN 9788370515805.
URL `http://iis.ipipan.waw.pl/2010/proceedings/iis10-10.pdf`

Wu, D. Y. *A comparison of approaches to determine topic similarity of weblogs for privacy protection.* 2011. [Online; last accessed 03-December-2012].
URL `http://dspace.sunyconnect.suny.edu/handle/1951/52388`

Xu, W., Liu, X., and Gong, Y. *Document clustering based on non-negative matrix factorization.* In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '03, pages 267–273. ACM, New York, NY, USA, 2003. ISBN 1-58113-646-3. doi:10.1145/860435.860485.

Appendix A

Supplementary Use-cases

This research focussed on evaluating the fundamental strengths of LSM with events. The work validates the use of the paradigm for Security Event Analysis and as a natural consequence, leads to further possible arenas of research. Table A.1 and Table A.2 contains use-cases considered for the research that are either derivatives of the use-cases that were already covered in section 3.4 or that have been earmarked as interesting but outside the scope of research due to time constraints or insufficient understanding of the method of applying LSM to the challenge. Some use-cases are related to Information Security but not to Security Event Analysis.

Table A.1: Supplementary Use-cases (1)

Use-Case	Derived From	Discussion
Detect New/Unknown Events	Use-Case 1	This use-case can be seen as the opposite of Use-Case 1 and would deal with events that could not be classified as being a specific or known eventtype. By virtue of not identifying an event as a certain eventtype, it would by definition be new or unknown and thus be cause for further investigation
Detect New/Unknown Sourcetypes	Use-Case 2	As with the preceding use-case, this use-case deals with events that could not be categorised as known sourcetypes. This use-case would be dependent on all events for known sourcetypes to be classified correctly before unknown sourcetypes could be reliably detected
Bad or Malformed Log Messages	Use-Case 1 and 2	Events that could not be classified or categorised may be malformed or bad; at the end of an iterative classification process, the user may be left with malformed or bad log messages
Whitelist or Blacklist Creation	-	Use an LSM map's co-occurrence matrix to inform most or least frequently occurring words, symbols or numbers. Experienced practitioners may use the data obtained to assist in the creation of blacklists, whitelists or other filters that could form a part of a SIEM implementation
Taxonomy Creation	Whitelist or Blacklist Creation	Use closely related words and phrases to inform the creation of a security taxonomy

Table A.2: Supplementary Use-cases (2)

Use-Case	Derived From	Discussion
Taxonomy Mapping	-	Train an LSM classifier using a security taxonomy. Each concept in the taxonomy would be mapped to an individual category. Attempt to match an event to a category in a taxonomy
Machine Type Identification	-	Train an LSM classifier using the results of machine inventory dumps such as a listing of software packages instead of events. Use LSM clustering to group 'configuration' documents into different categories. Categories should map to groups based on the contents of the configuration documents. Mail servers should be different from file servers and others
Machine Classification	Machine Type Identification	Use the LSM classifier derived from the preceding use-case and classify incoming machine configuration documents, possibly from scans happening on a daily basis to classify a machine as one of the known types
Misconfiguration Detection	Machine Classification	Machines not closely matching categories in the preceding use-case when measured by LSM Score may indicate that some variation exists in the software composition or versions
Miscellaneous Analysis Use-Cases	-	LSM can be used with any data; as such, the paradigm could be applied to more than events or configuration documents. Composition types could include packet data, protocol data and many other types of machine data that could occur within an IT Infrastructure
Conversation Clustering	-	In this use-case, the compositions could be instant messenger chat logs. It is possible that certain users or groups of users may discuss certain terms more frequently