2017

A Platform for monitoring of security and audit events: a test case with windows systems

Collins Chandi Kimathi
Faculty of Information Technology (FIT)
Strathmore University

A PLATFORM FOR MONITORING OF SECURITY AND AUDIT EVENTS: A TEST CASE WITH WINDOWS SYSTEMS

Kimathi Collins Chandi

Master of Science in Information Systems Security

2017

A PLATFORM FOR MONITORING OF SECURITY AND AUDIT EVENTS: A TEST CASE WITH WINDOWS SYSTEMS

Kimathi, Collins Chandi

Submitted in partial fulfilment of the requirement for the award of a Master of Science Degree in Information Systems Security (MSc. ISS) at Strathmore University

Faculty of Information Technology
Strathmore University
Nairobi, Kenya.

June 2017.

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student Name Kimathi, Collins Chandi

……

Approval

This dissertation of Kimathi Collins Chandi was reviewed and approved by the following:

Supervisor Name: Dr. Joseph Orero, PHD
Senior Lecturer, Faculty of Information Technology
Strathmore University

Dr. Joseph Orero, PhD
Dean, Faculty of Information Technology
Strathmore University

Prof. Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

Abstract

The rise in cyber attacks against organisations and government agencies has created a need for improving security and monitoring of Information Technology assets. Analysis and monitoring of security events are one of the key areas when it comes to detecting and preventing security compromises in any organisation. While intrusion detection and prevention are often used to measure security management in an organisation, there are challenges of false positives, false negatives and information overload to the analysts tasked with monitoring.

This work proposes to deliver an event collection and analysis system to monitor the security of Information Technology assets that have Windows Operating Systems, a centralised log management tool and dashboards to monitor analysed events in real-time for security alarms. The system will involve an agent to collect security and events from Windows Operating systems and send the events in a readable JSON format to the processing server for analysis and there after visualisation of various security events of interest. While security alarms such as bruteforce attacks can be identified and escalated to the security analysts. Testing was carried out by generating the desired security events from a Windows 10 virtual machine that were captured by the designed system.

Keywords

Log analysis, threat detection, application log, knowledge discovery, anomaly detection, real-time monitoring, log file format, Elasticsearch, Kibana, Logstash, dashboarding, alerting, windows security

Acknowledgements

The completion of this dissertation would not have been possible without the support of many individuals. First, I would like to express gratitude to my supervisor Dr. Joseph Orero for his guidance and patience during the process of writting this dissertation.
I would like to extend sincere thanks to my former colleaugues at Silensec for their input. My classmates Brenda, Stephen, Wilson, Rachel and Peris for the mutual encouragment. Rick and Morty for their remarks and motivation. Finally, to my family, who have supported me all through.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SIM – Security Information Manager.

SEM – Securiety Event Manager.

SIEM – Security Information and Event Manager.

NIST – Nattional Institute of Science and Technology.

OS – Operating System.

ACL – Access Control List.

HIDS – Host Intrusion Detection System.

NIDS – Network Intrusion Detection System.

CNP – Card Not Present.

PII – Personal Identifying Information.

CHAPTER 1: INTRODUCTION

1.1 Background

The security of servers and users' endpoint computers is important for many organisations, and being able to detect and/or prevent security incidents in computing systems should be key. This could be in line with an organisation's security and incident monitoring policy or just a best practice activity for the organisation. The threats against computing systems could range from malware activity, remote malicious user trying to gain access into the systems, unauthorised user access to files by real users who should not get the access and administrative activity being run by unknown users or non-privileged users among many other security risks (Tubin, 2013).

According to Manes Report (2016), Windows Operating systems are the most popular computing systems globally and hence many attacks are targeted and planned against these operating systems. These operating systems are not only used for personal computing but also for high level services in the IT infrastructure of organisations. Windows Operating System servers offer critical services such as Active Directory, Domain Control, Exchange Mail Server, File Server, Hypervisor, SQL Server, Web Server, DNS Server, and SharePoint Server among several more services that Windows server offers.

Windows Servers can also be used to host proprietary or custom applications that organisations need for their core business e.g. some banks have their core banking applications running on the Windows servers, while hospitals have their Hospital Management Systems running in Windows Servers. It is the extensive use of Windows Operating systems in this dangerous era of information security that calls for a research on how these events can be collected for the purpose of monitoring.

The collection of security and audit events is useful when it comes to forensic investigation but most organizations have no repository of all events that were carried

out during attacks. An attacker can delete the event trail from the compromised computer leaving no evidence of the malicious activities carried out. Having an remote machine to collect security events will be able to allow for better forensics investigation. This is because in most organisations there is generally a lack of proactive, comprehensive security systems dedicated to monitoring system irregularities (Glover, 2015).

There are few users with expertise on information security threat detection and while IT administrators are good at their jobs of system maintenance and management, very few are aware of the security problems that organisations could have. Organisations may not have the financial power to employ a security administrator who has studied with the skills and experiences on the Information security and hence it will have the risks that come with information systems (Magalhaes, 2003). Since management of security is not a revenue generating factor in IT and is often overlooked during budgeting making it a challenge for (Batchellor, 2016).

1.2 Problem Statement

Information Technology systems in any organisation will generate thousands of logs per minute, with security events being among them. A challenge is being able to identify and filter important security events that could be useful in detecting a security compromise or fraud. If one can collect the security events that are generated, and get real-time awareness of all the events that occur, then they will have a view of the organisation's security standpoint. With these data being analysed it is easier for an organisation to determine how compliant they are to regulatory policies related to information security (EY, 2014).

Current systems used for the extraction and analysis of security events face major challenges due to the enormous sizes of Windows event files and complexities in understanding the attack patterns connected to a security Incident. This leads to slow event analysis which is time consuming. Currently, forensic investigators and security professional lack efficient standard platforms to define and share attack patterns in regards to event analysis. Most of the existing solutions are expensive and cannot be affordable to most organiations while the opensource tools require a lot technical expertise to work effectively (RiskFocus, 2014).

1.3 Research Objectives

i. To review the role of security monitoring and event collection.
ii. To review the existing solutions used in security and audit and event collection.
iii. To design and develop and test a platform that can collect, analyse and visualise Windows security events.
iv. To validate the effectiveness of the proposed solution.

1.4 Research Questions

i. What is the purpose of security and audit and event collection?
ii. What are the existing solutions for security and audit and event collection and analysis?
iii. How will the platform be designed ?
iv. How will the platform be tested ?

1.5 Justification

The collection of security and audit events and logs to a remotes platform is one of the most common requirements for any information security compliance standard (Ipswitch,2010). This is because that an audit trail of all security events needs to exist outside the source of the events to ensure integrity and independence in case of a forensic investigation. According to the Guide to Computer Security Log Management by National Institute of Standards and Technology (NIST) recommends that all organisations are able to carry out audit log review for every system that is used to run the IT infrastructure as a measure of good practice (Karen Kent, 2006).

Having audit log review means that periodically the organisation checks to see was user activity of security impact has been carried out by mainly administrative users. This acts as a deterrent to those with malicious intentions and also aids in the battle against fraud carried out through computing systems. Some organisations may never know if unauthorised changes have been made to their Active Directory policies or Active Control Lists (ACLs), even in organisations where important Intellectual Property is stored in servers.

1.6 Scope and Limitations

The system developed is limited to a platform that collects and analyses events from Windows Operating systems. The scope uses Windows Operating System and services

that can offer security and audit events from the inbuilt Windows system. The research does not cover any other operating system apart from Windows based. The system is developed and deployed on virtual machines due to limitations in resources.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter explores the literature review with the purpose of identifying the need for security monitoring and event collection. It will give an introduction to the chosen Operating systems as a case for the study which is Windows Operating System. There will be a review of the existing solutions used for collection, analysis and visualisation of Windows security and audit events.

2.2 Security Threats in Operating Systems

A computer is both a logical and physical system, the hardware part is what we all see but what we do not see is the logical part. The logical part involves software that uses the hardware resources to allow us to use the computer (Rouse, 2013). But before software is made useable to the people there needs to be an interface that connects the hardware to the software and that is the role of the Operating System. The Operating System (OS) is a program that manages other programs, in this case application programs, it is the role of the Operating Systems to ensure that physical resources provided by the hardware are useable to the applications and the user, such as: allowing for multitasking where multiple applications can run concurrently as the Operating System manages the use of CPU and RAM to avoid overloads. This means that it will allow the input and output hardware i.e. keyboard, mouse or monitor to be useful and also manage the resources such as RAM, ROM or Processor as they are used by the application programs (Rouse, 2013).

Threats to Operating Systems are those that create unauthorized actions by users of a computer (Microsoft, 2011). These extended to include the breach to privacy, confidentiality, and integrity. Threats are categorized as either malicious or non-malicious. The non-malicious attacks are those that come from users and employees with no proper training on computers and have no awareness to security threats that

computers are predisposed to. On the other hand malicious attacks usually come from external entities or disgruntled employees who are well aware of their actions as they compromise the security of the organization (Rouse, 2013). Information has become a valueable asset in most organizations and to gain a competitive edge most attackers have the aim of acquiring trade secret information from companies. Since Operating Systems are the base of every computing system they are easily targeted for security compromise. Attackers will try to force authentication and exploit vulnerabilities in oprating systems so as to collect information processed by applications or stored within the computer. One of the most common way to compromise Operating Systems is by compromising the password of a user. The problem usually is remembering the correct password from among the multitude of passwords a user needs to remember, this leads to password reuse becoming common. Attackers know that people are the weakest link to security and having phishing attacks to aqcuire passwords can become fruitful if a user has reused the password in a computing system (Rouse, 2013).

2.3 Windows Operating System

Windows is a line of operating systems developed by Microsoft and it is the most popular globally with over 80% market share (Hruska, 2016). This because of its flagship OS with a graphical user interface (GUI) which was Windows 1.0 in 1985. Having the GUI allowed computers to be more useable to everyone, including those who were not skilled in MS DOS (Darcy, 2014). Over the years Microsoft has improved the Windows OS with newer versions that have still dominated the market (Hruska,2016). The most current one is Windows 10 for personal computers which was released in 2016 and for servers, there is Windows Server 2016 (Chapple, 2016).

2.4 Security Management Systems

2.4.1 Security Information Management (SIM)

According to Kumar (2012), Security Information Management (SIM) is the practice of,

collecting events from multiple security log sources, monitoring, analysing security events and having a long term event storage plan for the events.

A SIM system will automate this process, making it simpler for the security team to have a central collection point for security events from systems such as Intrusion Detection Systems, firewalls, proxy servers and anti-virus and other systems within the organisation. A typical SIM should be able to collect events from various log sources, normalise all collected events into a standard structure, then analyse the data to give a proper presentation of all security events from the infrastructure (Schmidt, 2013).

A SIM will allow for log collection of security events by normalizing and indexing the data then compressing the events from raw unstructured data into structured information that can be used for trend analysis or compliance reporting. Event collection happens in several ways depending on the source of the event. This can be done from configuration on the application or operating system and when that is not possible an agent application can be installed to collect and send the events to the SIM server (Kumar, 2012).

2.4.2 Security Event Manager (SEM)

Security Event Managers (SEMs) have a different purpose from SIMs as they are used to collect events, normalise it, then use correlation to generate alerts and alarms that will be resolved or analysed by the security analyst (Kumar, 2012). The SEM will summarise the events into only important information and will not store the raw logs unlike the SIM . After correlation of events and filtering of security rules the security alerts generated by the SEM be assigned to various administrators as tickets to be resolved. Most SEMs have reporting capability but they are not as customizable as those from a SIM due to the amount of data stored by the SEM from collected events. The stock reports could be a collection of compliance reports of things such as failed authentication, account lockouts or antivirus detections in the course of the month. SEMS will generally provide (Schmidt,

2013) the following: event management, real time threat analysis, visualisation, ticketing, incident response.

SEM products are ideal for running security operations such as a Managed Security Service Provision (MSSP), this is where an organization does the security management and monitoring for several organizations. The major weakness for SEM is that they are not good at log management, long term storage or log compression unlike SIM products (Kumar, 2012).

2.4.3 Security Information Event Management (SIEM)

A "SIEM" is a group of complex technologies that together provide a bird's-eye view into an infrastructure (Pigee, 2016). It will combine the capabilities of SIM and SEM, to give real-time analysis of security alerts generated by the systems in the infrastructure, while allowing for long term storage of logs captured. The SIEM should gather, analyse (including correlation) and present information from the systems in the infrastructure. This information could be from: access control information from OS, application and devices, vulnerability management tools, policy and compliance tools, database and application logs (Kumar, 2012).

The SIEM has multiple features that enable it to be as versatile as it is. The main feature of a SIEM is Event Collection. A SIEM should first be able to collect events from all possible log sources including custom application (Pigee, 2016). Once events have been collected the next step is Log Normalization here the SIEM should organise the data into respective fields, think of this like in a database table where a row contains a single event while the columns represent different fields on each row such as username, Event ID, log source, IP-address, timestamp and so on (Pigee, 2016). Indexing happens once the data has been normalised, the indexing of important data from the database should happen which allows for faster searching and correlation with the important fields such as usernames and IP addresses (Tenable, 2015). Data aggregation of the event logs coming

into the system allows for similar events to be consolidate so as to increase efficiency and save on storage (Kumar,2012).

Analysis is done by the SIEM through inbuilt filters and event rules to identify events that are deemed to be of security concern to the organisation. This could be events such as remote logins to servers or network scanning packets being identified by the IDS. After the analysis of events correlation can be done on the events. Correlation is where two or more different events have a consequential outcome, for example when multiple failed logins occur and then one becomes successful under a span of a few seconds, this could be a sign of brute force attack against the system (Pigee, 2016).

With a SIEM, alerting can be done once correlation and event analysis has been completed and parametres set by the security analyst. The SIEM should create or send alerts via email, SMS or any other configured mode of notification to the relevant parties (Kumar,2012). Dashboards allows for visualisation of events of different categories. The dashboards are mainly high level view of the events but can also be technical e.g. when showing port statistics (Kumar,2012). Compliance of an organization to certain security standards can be done with all the events that the SIEM is able to collect from the various log sources. The SIEM should be able to give a view of certain compliance checks and then give reports that will answer some compliance checks e.g. ISO 27001, PCI DSS, HIPAA or any other regulatory standard (Kumar,2012). Retention of logs is a feature that the SIEM needs in order to ensure that investigations can be carried out properly if need occurs. The retention period in most cases should be at least a year worth of events collected (Kumar,2012).

2.5 Host Intrusion Detection Systems (HIDS)

These are applications that are installed in servers and end user computers to carry out to do log analysis, event correlation, integrity checking, policy enforcement, rootkit detection, and alerting then send normalised events to other security devices (SANs

Institute, 2000). In Windows they will read the binary files in the Windows event manager and forward security events to the SIEM. These events could be like user being created or deleted in the operating system, password reset or changes and other security and audit events that could be of interest to security analysts. On UNIX systems the HIDS will read the auth.log file to capture similar events that are of security impact to the operating system (Bray, Hay, & Cid, 2008).

When it comes to integrity checking, the HIDS will run a MD5 checksum on all critical files in the operating system and then monitor the file regularly for any change in the MD5 checksum value. If a critical file e.g. a Windows registry file is modified they event is sent to the SIEM where an alert is generated to the security analyst (Bray, Hay, & Cid, 2008).

2.5.1 Challenges With Security Monitoring Systems

According to IT auditing software company Netwrix Corporation, 69 per cent of companies are looking to reduce SIEM bills while 65 per cent of users calim that it is hard to find necessary data upon request in a SIEM solution (Netwrix, 2016). The survey done Netwrix Corporation targeted 234 large organizations that used SIEM solutions for security and IT infrastructure monitoring. The results showed that SIEM solutions become expensive when it comes to their maintenance and support. While the cost of owning a SIEM was also driven up by the need to hire and train SIEM analysts (Fadilpasic, 2016).

The top problems with SIEMs according to Alienvault (2014) is that, they are too complex especially for organizations to deploy in order to adequately claim to have security monitoring. The deployment of SIEMs usually takes too long as the SIEM has to be configured over a lengthy period of time in order to reduce false positives and fine tune as per the organizations needs. The licensing model to acquire a SIEM are also expensive while deployment consultants have to be paid to ensure that the software bought as a

SIEM becomes functional as expected (Alienvault 2014).

2.6 The Intrusion Detection Challenge

Security has become a key concern for developers and creators of software and any IT system, this is has been brought about by the increase in security incidents on IT systems over the past few years. With weaknesses in confidentiality, integrity, availability and many others being exposed on IT systems the need for security being a key integral part for any software or electronic system has increased.

It is the confidentiality weaknesses in systems that has allowed many users' data to be exposed and used for malicious activities such as black-mail or identity theft. This has happened when systems do not store Personally Identifiable Information (PII) properly leading to exposure of patients' medical records from hospital systems and credit card information from e-commerce systems. According to The Data Breach Report by ITRC (2015), over 169 million personal records were exposed and that number is only from 781 breaches that were made public by the affected institutions from the financial, education, government and healthcare sectors. With an IBM report giving the average cost of each stolen record with confidential and sensitive data being $158, while those records

collected from the healthcare sector having a price for $355 each (Ponemon, 2016).

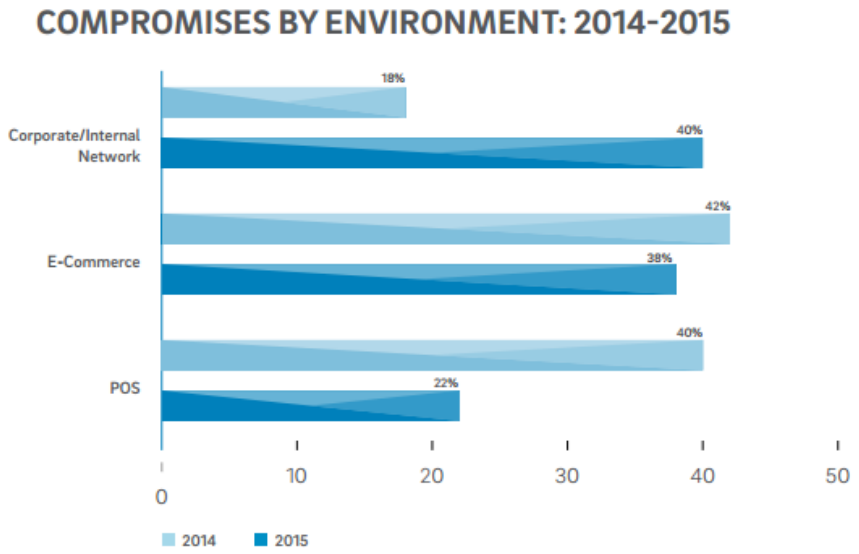

Figure 2.1 Security Compromises (Trustwave, 2016)

In 2016 only 41% of breaches were detected by the victims with the median time between the intrusion taking place and the victim being able to detect the intrusion being 15 days, while those detected by an external party being 28 days (Trustwave, 2016) . This shows how much time is taken for an attack to be detected by the victims, one can only imagine what kind of control an attacker can have after 15 days of compromising an organisation's IT infrastructure without being detected. The biggest growth in compromises in 2015 was in the Corporate and Internal Network of organisations as shown in figure 2.1.

Figure 2.2 shows that most of the attacks in 2015 were mainly targeting organisations and users, with destructive intents and card-not-present (CNP) being the most popular types of data compromise.

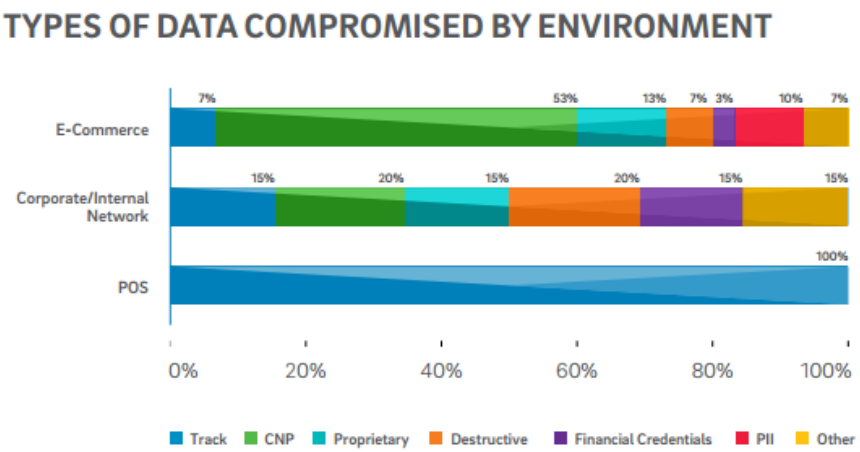

Figure 2.2 Types of Data Compromises (Trustwave, 2016)

2.7 Compromise Duration

According to the Trustwave Global Security Report (2016), when it comes to an IT security incident, there are three key milestones in an attack lifetime, these are:

a) Intrusion: the date where an attacker was able to gain unauthorised access to the victim's systems.

b) Detection: the date on which the victim was able to identify the breach had taken place.

c) Containment: the date when the compromised was stopped and the attacker denied access to the victim's system.

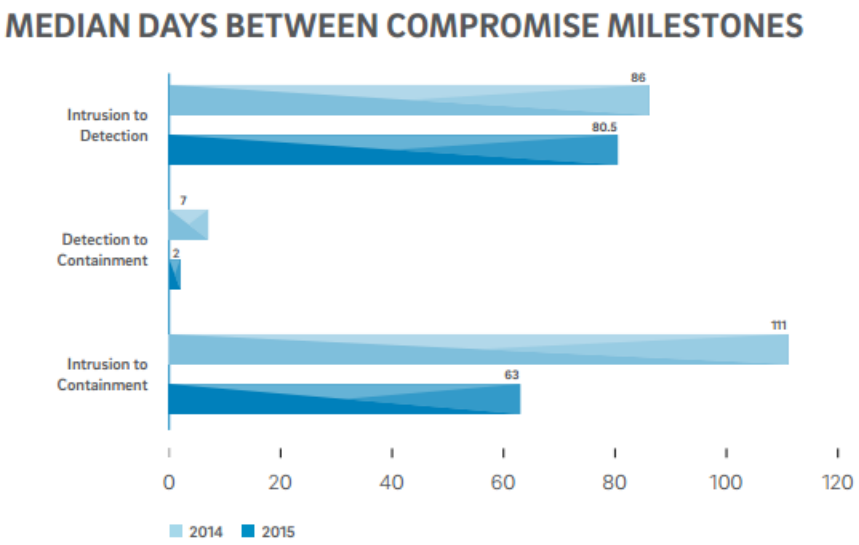

Figure 2.3 Detection Time Span (Trustwave, 2016)

In 2015 it took a median of 80 days for the victim to identify that they had an intrusion had taken place in their environment. It took 2 days for after detection for the containment to occur and 63 days from intrusion to containment, where containment was possible. From these one can clearly see that 80 days of being compromised without the victim being knowledgeable of the situation and even having a more challenging time containing the intrusion. It is this point that has shown a gap in early detection of security incidents.

2.8 Purpose of Event Logging

Most Operating systems and enterprise applications are designed with a logging feature, primarily it was used for debug information to show system administrators events that had led to an error. Developers used logging to catch exceptional conditions and log them to support staff, read invalid values and check if the applications work in the correct order (Kumar & Nanda, 2001). The option for security event logging was always there, although security was not a major issue in the early 2000 where bugs and programming

errors were of greater concern (Sugano, 2003). The major issue seems to have been availability of the systems which was an issue because of programming maturity was not as it is today. Today logging of security and audit events is more or less a requirement from a compliance and governance standpoint (Ipswitch, 2010).

2.9 Event and Log Management Best Practices

Having and a defined audit policy is the first step for the organisation, the policy defines what type of events should the applications and operating systems record. Generally most compliance standards have similar requirements for an audit policy. In Microsoft Windows environment the audit policy can be configured from the domain controller in a LAN by modifying the group policies to reflect the recording of the required events. From the group policy settings the administrator can set different event logging settings for individual servers depending on their use and also for workstations (Microsoft, 2011). E.g. in servers that have MSSQL database running one will collect audit events for database access which may not be possible for servers that run Mail Exchange services, while on user workstations one can create policies depending on users; the CEO audit policies could be more serious compared to those of an intern since the type of information in the CEO's machine is highly sensitive (Ipswitch, 2010). Events logging categories that must be enabled include the Success and Failure of logon events, object access process tracking, policy change, account management, directory services access and systems events.

Another best practice for event and log management is having a centralised collection and processing system (Lewis, 2014). This means that all events from production servers and users' workstations are collected in real-time to a server within the organisation or to a remote area. The system collecting the events should also be able to process and parse the various events into useful information. In most cases the events are only hosted in the generating machine, meaning that a malicious user can delete the events from the logging

system once they have finished their activity (Lewis, 2014). Having agents installed in each system is necessary to collect events as they are generated and shipped to the central log server. Some compliance standards such as Sarbanes Oxley (SOX) require that the logs to be stored for a year or even up to 7 years (Netwrix, 2016). Most systems may not have the capacity to store their own events for more than a year without having to write over old events for disk space management. Also the servers maybe replaced by newer once after a few years of usage, which means that if a server in a forensics case was retired before the statutory limit was over without having its events stored remotely, then the case will miss the evidence that the server could be holding. Having a centralised copy of events means that the logs can be archived and retrieved whenever needed. The data being retrieved must be of a reliable source with high integrity and having a systems that does event collection automatically is better than what any number of human beings can do.

Once meaningful information is processed in real-time one can create filters that will define which events should trigger alerts e.g. when a new user is created in the Active Directory. Reports are also integral to event collection as they can give high level to detailed information on the security status of the organisation, they are also useful for stakeholders such as management, auditors, security and compliance officers (Ipswitch, 2010). The most common events to be monitored and alerted on in Windows systems are: changes to access control lists (ACLs), registry access and modification like additions and deletions, user account changes such as users getting administrative permissions, active directory access and changes, changes to groups – adds, deletions and changes, multiple windows login failures, successful logins, locked accounts, password change, password reset, application start, failure or shutdown, server reboots, dns changes, system events such as process start and stop, web server access and permission changes, ftp server access and file transfer, policy changes in the domain controller.

2.10 Compliance and Standards on Event Collection

2.10.1 Sarbanes – Oxley

In the USA a mandatory act was passed to ensure corporate governance and regulation of financial practice. The bill was created by Senator Paul Sarbanes and Representative Michael Oxley in 2002. It is a regulations that must be complied to by all organisations in the USA. The Sarbanes-Oxley Act of 2002 (SOX) is used to protect investors from fraudulent accounting activities, it was created as a response to the Enron Corporation and WorldCom organisations were found to be carrying out fraudulent bookkeeping (Investopedia.com, 2003).

The requirement that for Management Assessment of Internal Controls has been set by the SOX guidelines in Section 404. This section requires that all annual financial reports must include an Internal Control Report stating that management is responsible for an "adequate" internal control structure with an assessment of the effectiveness of the control structure being included within the report. External auditors should validate the accuracy of the reported internal controls ("Sarbanes-Oxley act summary of major sections," n.d.). According to the Ipswitch white paper on Best practices on event log management (2010), when it comes to Information Systems the following are what standard requirements for the SOX regulations are: Identification: Logging and reporting on all user identities and access including privilege users to ensure that all users can be uniquely and irrefutably identified (Ipswitch, 2010). Authentication: All transactions from systems that can authenticate users should be logged and reported (Ipswitch, 2010). Policy-based access control: log and report that only authorised business users have access to systems, data and network assets (Ipswitch, 2010). Data Protection & Integrity: log and report on access to data, who accessed data, how long and if data was changed, modified or copied, data integrity fed from upstream sources into the application system (Ipswitch, 2010). Identity provisioning: Log and report of access for all users including time-specific restrictions or access control based on the location of the originator

(Ipswitch, 2010).

2.10.2 HIPAA

Health Insurance Portability and Accountability Act of 1996 (HIPAA) ensures that confidentiality, integrity and privacy of healthcare information stored and transmitted in electronic form from a information security standpoint (HHS, 2015). HIPAA provides strategies to protect against threats to the security and integrity of patient information but also requires that organisations have records sufficient enough for an audit trail in case of a breach. The audit trail must be sufficient enough to establish what events happened, when they happened and who/what caused them (Ipswitch, 2010). For HIPAA the requirements for event and logging are from Security rule 164.306 and Privacy rule 164.530. On Windows Systems the possible audit and security events required for logging are: password aging, consolidated change logs, user privileges, ntfs permissions, system privileges, role permissions and membership, remote access, user access, auditing enabled.

2.11 Windows Event Analysis

2.11.1 Windows Logging

In the Windows environment logs are generated in relation to the operating system, applications and security. Application log which are events logged by applications such as IIS, MSSQL, DNS, Domain Controller and other Microsoft applications (Magalhaes, 2003). Security log which contain show logon attempts which could be successful or failed authentications as well as events related to resources use, such as creating, opening, or deleting files or other objects. System log contains system component event such as driver failures and hardware issues (Magalhaes, 2003). In Windows Operating Systems the Security Logging option is by default not turned on, and it requires the administrator to enable it from the logging setting (Magalhaes, 2003).

Windows log messages are known as Events and are stored in the "C:\Windows\System32\winevt\Logs" directory in binary format (Forensics Wiki, 2016). Windows XP, 2000, 2003 store logs with .evt extension whereas newer Windows versions use the .evtx extension(Forensics Wiki, 2016). All Windows event logs can be viewed using the Windows native Event Viewer.

2.12 Analysis Using Windows Event Viewer

The Event Viewer is a Windows native graphical application used to view Windows binary logs (with extensions .evt or .evtx), which is available on most of the Windows versions. Events are displayed based on the LogName, which is just a reference to the logs generated by a specific application such as Application, Security, System, Setup and etc (Stanek, 2000).

Analysis of windows events can be done in several ways, the first one and most common is using the Windows Event Viewer. This is an application that will access the Windows Event Log files which are stored in the directory C:\Windows\System32\winevt\Logs and display the events in a formatted view (Microsoft, 2011). According to Hoffman (2012) the Event Viewer can help administrators find errors to help troubleshoot problems in applications. The Event Viewer will display events in specific categories such as Application log for application events or System log for Windows system events (Hoffman, 2012). The event viewer will show events of the following type for each category of logs captured.

Table 2.1 Event Types in Windows (Microsoft, 2011)

Type	Description
Error	Events that indicates a problem such as data or function loss. If a service stops running (Stanek, 2000).

Warning	This events are used to indicate a future problem that could arise e.g. disk running low (Stanek, 2000).
Information	This denotes events that are from successful service activities such as a service running after a restart (Stanek, 2000).
Success Audit	This shows events which are successful security access attempts e.g. a successful attempt to login (Stanek, 2000).
Failure Audit	This is for a security access attempt that fails e.g. a user with wrong credentials attempting a login (Stanek, 2000).

According to William Stanek (2000) events on the event viewer, it will also contain information about the Source as the application, service, or component that generated the event. The category of the event to further describe the related action. An EventID as identifier for the specific event. The user account name that was running the service or application. The name of the computer where the event occurred. A text description of the event and any data or error code output by the source about the event.

The Event Viewer also allows for other Windows based systems to forward their events to a central Windows server dedicated for event storage which can be useful to monitor several Windows Systems from a central Event Viewer (Costea, 2016). The challenge here is that only Windows Servers can send logs to the Event Viewer limiting if the organization is to expand event collection to UNIX or network devices. The Event Viewer will provide the IP address of important security events that can be useful during investigations. The challenge is that IP addresses cannot be filtered using the mechanism provided by the Event Viewer (Mullinix, 2013). According to Microsoft Support (2015) the Event Viewer with newer version of Windows can handle a maximum of 16GB for total size of event logs, which is not very sustainable when it comes to enterprise archival.

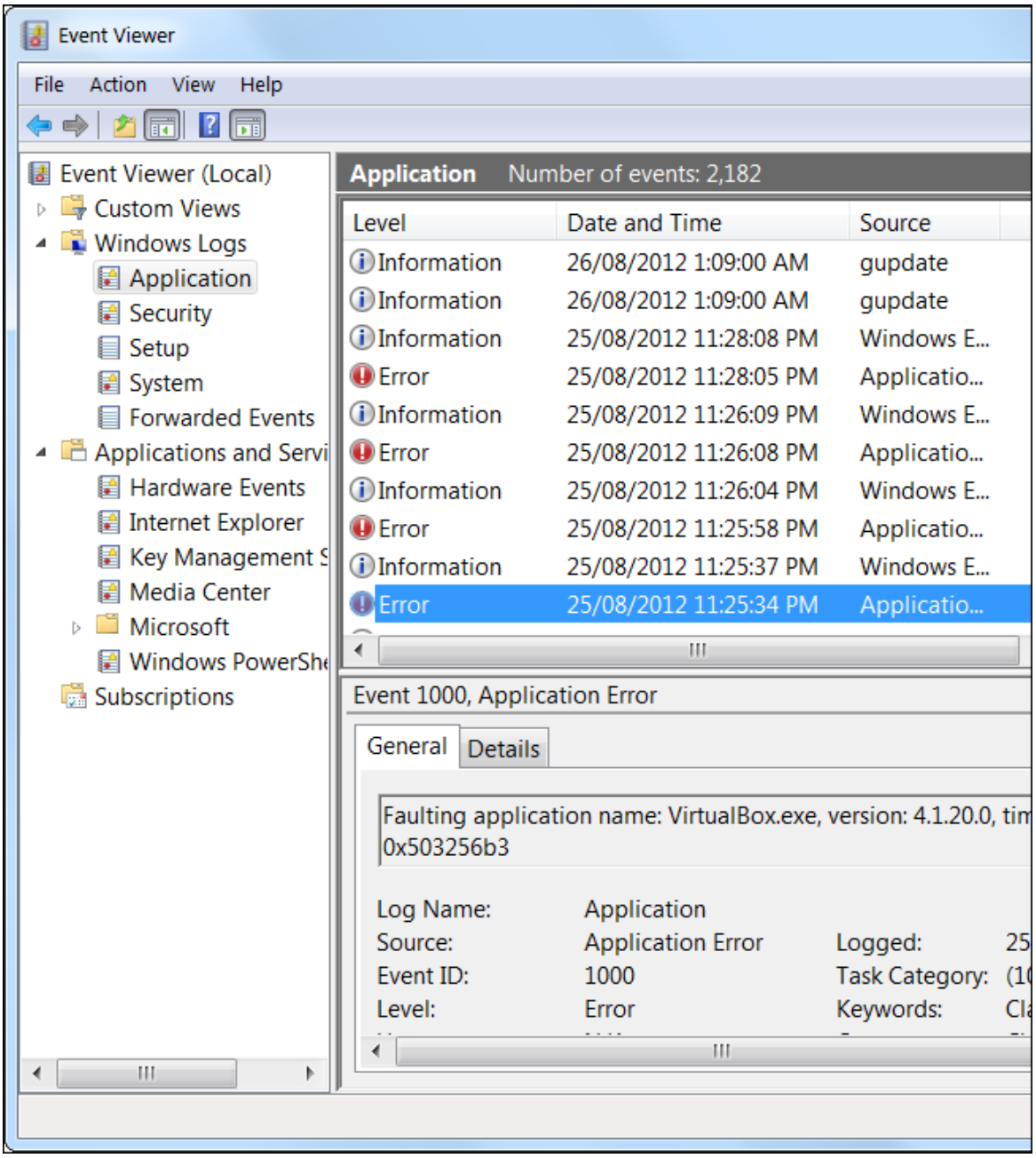

Figure 2.4 Windows Event Viewer (Hoffman, 2012)

To analyse events from the Event Viewer one has to create filters for specific events which is best done by knowing the EventIDs for each security events. The most common security EventIDs to be analysed and reviewed are:

Table 2.2 Windows Security Event (Hoffman, 2012)

Event Name	Event ID
Account Lockouts	4740
User Added to Privileged Group	4728, 4732, 4756
Security-Enabled group Modification	4735
Successful User Account Login	4624
Failed User Account Login	4625
Account Login with Explicit Credentials	4648

2.13 Analysis Using Regular Expressions (Regex)

Regular expression is the use of strings with character literals and operators that can be used to identify specific patterns from text data (Rouse,2016). According to Russ Anthony (2013), event filtering from the windows Event Viewer can use regular expressions to capture various events as seen in Table 2.2. Jamie Zawinski the creator of Netscape once said "Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems" in reference to the difficulty of using regular expression (Zawinski, 2011). The challenge with using regular expressions is creating strings to effectively parse a different types of events without missing key elements in each. This can be resolved by having a developer modify or add regex filters until all necessary events are captured by the analyser properly.

Table 2.3 Regular Expressions for Windows Events (Anthony, 2013)

Windows 7 regular expressions	**SOURCE**	**EventID Number**

".*APPCRASH.*"	Application	1001
".*he protected system file.*"	Application	64004
".*EMET_DLL Module logged the following event:.*"	Application	2
".*A new process has been created\..*"	Security	4688
".*A service was installed in the system\..*"	Security	4697
".*A scheduled task was created\..*"	Security	4698
".*Logon Type:[\W]*(3|10).*"	Security	4624, 4625
".*\\Software\\Microsoft\\Windows\\CurrentVersion\\Run.*"	Security	4657
".*service terminated unexpectedly\..*"	System	7034
".*service was successfully sent a.*"	System	7035
".*service entered the.*"	System	7036
".*service was changed from.*"	System	7040

2.14 Microsoft Azure Log Analytics

Microsoft Azure is a public cloud computing platform developed by Microsoft to host various products and services over the cloud such as computing, storage, network and analytics (Rouse, 2016). To collect events from Windows Systems the Azure offers a Log Analytics service will collect events through the Microsoft Monitoring Agent (MMA), which will send the collected events to the Log Analytics service hosted in Microsoft

Azure (Anderson and Rundle, 2017). Microsoft Azure Log analytics is a closed source service where users are limited to only the options that are supported by Microsoft Azure, this means that having custom events or application events that may be found in other sources is a challenge while it also has a limited visualizing element (Cavanagh, 2016).

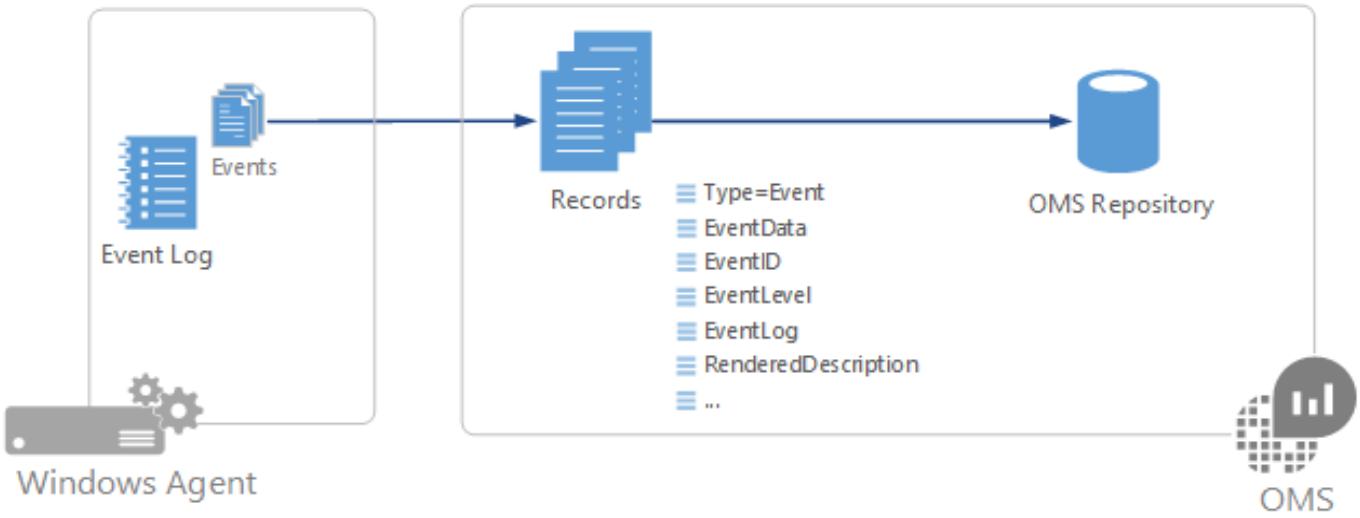

Figure 2.5 Azure Log Analysis Service (Anderson and Rundle, 2017)

The Log Analysis service in Azure will process the events sent by the Microsoft Monitoring Agents into the following parts (Anderson and Rundle, 2017):

Table 2.4 Event Analysis with Azure Log Analysis Service (Anderson and Rundle, 2017)

Property	**Description**
Computer	Name of the computer that the event was collected from.

EventCategory	Category of the event.
EventData	All event data in raw format.
EventID	Number of the event.
EventLevel	Severity of the event in numeric form.
EventLevelName	Severity of the event in text form.
EventLog	Name of the event log that the event was collected from.
ParameterXml	Event parameter values in XML format.
ManagementGroupName	Name of the management group for System Center Operations Manager agents. For other agents, this value is AOI-
RenderedDescription	Event description with parameter values
Source	Source of the event.
SourceSystem	Type of agent the event was collected from.
TimeGenerated	Date and time the event was created in Windows.
UserName	User name of the account that logged the event.

2.15 Analysis of Windows Events using OSSEC

Ossec is a Host Based Intrusion Detection System (HIDS) that relies of rules to analyse security events that its agents collect (Bray, Hay, & Cid, 2008). According to Daniel Cid (2007) in Log Analysis using Ossec, "Log Analysis for intrusion detection is the process or techniques used to detect attacks on a specific environment using logs as the primary source of information". When it comes to Windows Security events analysis with ossec, an agent will be installed in the Windows Operating System as it is the primary source of information. The Ossec agent will collect all relevant logs and send them to the Ossec server for analysis. Analysis at the Ossec server is done by decoders based on Regular Expression and rules that are able to prioritize on events before alerting (Bray, Hay, & Cid, 2008).

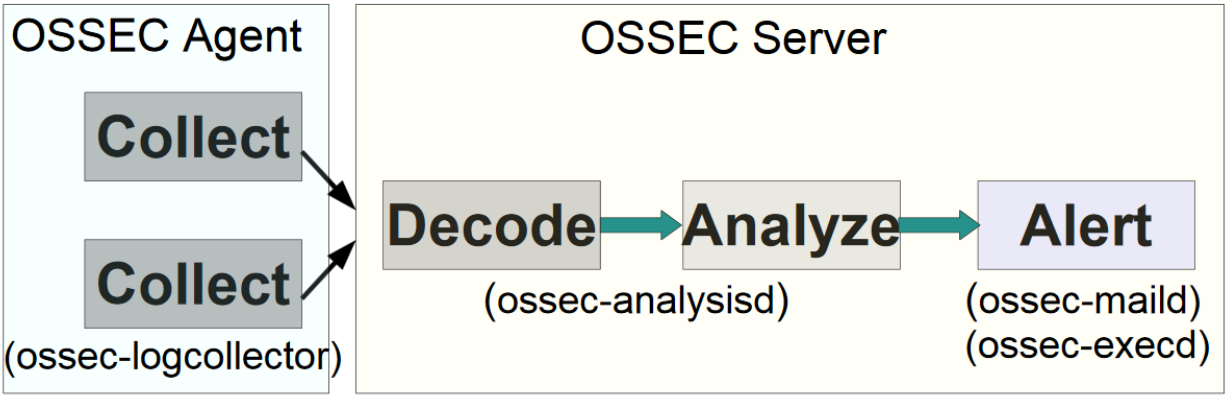

Figure 2.6 Ossec Event Analysis (Cid, 2007)

2.15.1 OSSEC Event Analysis Process

The log analysis at the Ossec server works in three main parts (Cid, 2007):

- Pre-decoding – Extracting known fields e.g. time.

- Decoding – Extracting unexpected fields such as usernames, ip addresses.
- Signatures – Using defined rules to prioritize and alert on events.

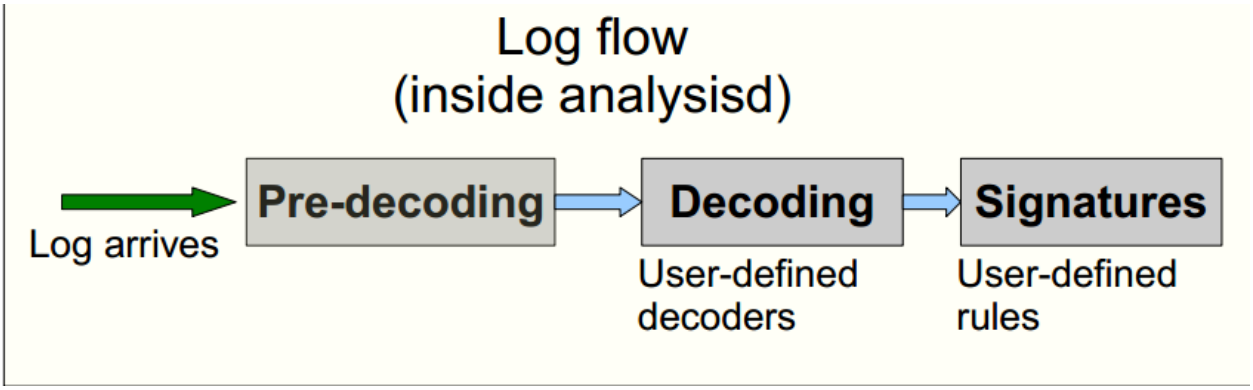

Figure 2.7 Ossec Internal log flow (Cid,2007)

A pre-decoder for windows date:

```
<decoder name="windows-date-format">
<prematch>^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d </prematch>
</decoder>
```

An example of a decoder for windows events:

```
<decoder name="windows1">
<type>windows</type>
<parent>windows</parent>
<regex offset="after_parent">^\.+: (\w+)\((\d+)\): (\.+): </regex>
<regex>(\.+): \.+: (\S+): </regex>
<order>status, id, extra_data, user, system_name</order>
<fts>name, location, system_name</fts>
</decoder>
```

An example of a signature for windows events:

```
<rule id="18110" level="8">
<if_sid>18104</if_sid>
```

```
<id>^624$|^626$|^4720$|^4722$</id>
<description>User account enabled or created.</description>
<group>adduser,account_changed,</group>
</rule>
```

The challenge with OSSEC is that it currently lack a method for visualizing events captured and analysed as the OSSEC Web User Interface was stopped as of 2013 and is no longer maintained by its creators (Parriot, 2013).

2.16 Log Analysis Tools

Log analysis consists of: parsing fields in collected log records, the ability to view the records in a consolidated form, perform search operations for specific data using custom queries and highlighting results that might be of interest. While the log file viewers with searching capabilities might offer sufficient functionality for basic application monitoring, they may not be good enough when it comes to the collection logs from multiple sources with high volumes. The event analysis and searching in high numbers of log files records for specific issues will often require more resources and applications built to handle large amounts of events, with the ability to store them for archival purpose. Searching and filtering is often based on regular expression input and configurable queries filtering contents, which is also a limitation to evnt log file viewers such as Microsoft Event Viewer. Below are application log files view and analysis tools:

Log Expert – This is a free open source tool for Windows, contains search, filtering, highlighting and timestamp features (Raab, 2011).

Chainsaw – This is an open souce desktop application for Apache logging services focuses on GUI-based Log4J files view, monitoring and processing. It offers searching, filtering and highlighting features (Apache, 2007).

Retrospective – Commercial solution for managing log files data working on multiple platforms and offering wide search, monitoring, security and analytic capabilities with a

friendly UI design. Pricing for personal use starts at $92 (centeractive, 2015).

2.17 Centralized event logging and monitoring systems

The key factors to consider when choosing an event logging and monitoring system are as follows:

Visualizations – This means being able to create graphs and charts of information that is collected (Warda, 2017).

Accessible everywhere. – This is remote accessibility of the systems, this is best achieved by having a web accessible system (Warda, 2017).

Remote location. – Having the system run in it's own dedicated resources rather than it residing on the same location as the source of the monitored events (Warda, 2017).

Centralized logging. – Having a core location where all event logs from multiple machines are sent automatically and stored without having to collect them on demand (Warda, 2017).

Custom events. – The system should be flexible enough to allow custom events from unknown sources to be stored and analysed (Warda, 2017).

Real-time. – Having events being collected and analysed in the system as they happen on the machines generating the events (Warda, 2017).

Alerting – The ability to include alerting mechanisms for rules and thresholds that one may want (Warda, 2017).

Historic data. – Having the ability to keep collected events for a specified retention period as demanded by the user (Warda, 2017).

2.18 Conclusion

Computer systems, networks and software applications events are recorded in their log files. Log files serve as an important source of information for purposes of analysis of security breaches in the system. Since most systems maintain their events in log files, this

is beneficial in identifying problems and security threats in the system through analysis of log files for pattern identification indicating suspicious system behaviour. In the past, log file analysis was carried out manually which would lead to missing some event logs containing important information. Log files are large in size and this would prolong the process of log analysis. Having a platform to carry out log analysis will enable security events to be identified with more ease. This will eliminate the time taken for compromise to be identified and swift action by the incident management can be followed through before damage is realized by the organization. Customizable visualizing of these events will allow analysts to identify patterns and anomalies more effectively as compared to having manual analysis of events.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

This chapter deals with the software methodology to be used in the development process of the system. The software methodology was made up of the following key sections: Requirements analysis, system design, coding, testing and evaluation methods then implementation.

3.2 Software Methodology

The software methodology used for the developemt was Secure System Development Life Cycle (SSDLC). It is a multistep method that starts with system analysis, system design, implementation and testing; it then stays through the maintenance and usage of the system. SSDLC is the most effective methodology to secure information and information systems because it integrates security into every step of the system development process, from the beginning of a project to develop a system to its disposition (Crosby, 2016).

Figure 3.1 Secure SDLC (Crosby, 2016)

3.3 Requirements Gathering

This is a phase where user requirements were identified and analysed. The analysis results were used in answering the research questions and also used while designing and developing the proposed system.

3.3.1 Feasibility Study

This was conducted through literature review on existing and other proposed methods used for windows event analysis.

3.3.2 Research Design

The kind of research design to be used greatly relies on the specific research objectives and has to relate with the research questions (Kothari, 2004). The quasi-experimental research design was used for this case study. Quasi-experimental research designs test the relations in given environments with the aim of analysing outcomes of interest based on treatments. This research design was chosen as the research analyses if the proposed event analysis approach is able to identify security concerns on Windows Operating Systems and accurately analyse them in a timely manner.

3.4 System Design

System design involves the sectioning of the system to be developed into components for purposes of studying how these components work together to achieve system functionality (Gemino & Parker, 2009). The use case diagram models the system functionalities and gives an illustration of how system actors interact with the system processes known as use cases (Mishra & Mohanty, 2012).

3.5 System Implementation

The development of the system involved the construction of a web vizualization application and an event analysis system that is connected to a non relational database. The system holding the applications has the following specifications all under a

virtualised environment. Below are the approaches that were used in the development of the Analysis and Event processing System

i) Data Visualisation System

The web application was developed using Kibana, which is a visualisation application for the Elastic Stack, it provides a user interface for Elasticsearch giving visualisation in terms of graphs, charts and other visual mappings of data (Anderson & Kuć, 2016).

ii) Database and Indexing System.

The database was developed using ElasticSearch database, which is a non relational database (Banon, 2010). Reasons for using ElasticSearch are: it is an open platform, it is fully compatible with PHP and other platforms and it is a non-relational database that does indexing of all data entered this is for the best performance or a searching platform (Cholakian, 2014). Elasticsearch is implemented in RESTful HTTP/JSON using a java library called Lucene, which allows for a schema-free database server which uses different terms from relational databases (Khan, 2013). The comparison between elasticsearch componenets and relational databases is as seen in Table 3.1 below.

Table 3.1 Comparison between Relational Databases and Elasticsearch (Khan, 2013)

Relational Database	**Elasticsearch**
Database	Index
Table	Type
Row	Document
Column	Field
Schema	Mapping

iii) Event Analysis System

The event analysis engine was developed using Logstash which is a data processing engine that enriches data from any kind and gives it a proper context (Anderson & Kuć, 2016). Logstash was configured to parse Windows events that are sent from the agent

using a json parser. Logstash is good at extracting useful information from bulk data that comes into the system and normalising the data to useful fields that are then sent to Elasticsearch for storage and indexing (Anderson & Kuć, 2016).

iv) Data Collection Systems

A windows event collection agent was used to collect events from windows systems. These are agents used to query and collect data from Windows Operating systems sources and send the captured events to a system for processing. The agent used was Nxlogs community edition, which is a small and easy to deploy for whatever location they are needed in.

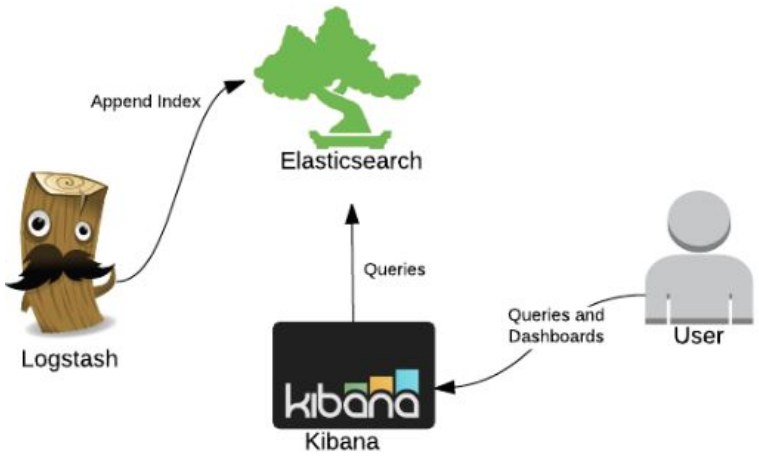

Figure 3.2 Elasticsearch, Logstash and Kibana Setup (WiredPulse, 2015)

3.6 System Testing

The system was tested against its specifications to verify whether it complied with the functional requirements. The following tests were done:

Unit Testing: This was to test if the individual componenets of the system were able to function as required. This was done by checking the service status of each service in the system and monitoring for unexpected outputs.

Compatibility Testing: Since the system is accessed from only the web browser, the test was done to find out if a user is able to access the web application from the following web

browsers, Google Chrome, Mozilla Firefox and Internet Explorer.

Functionality Testing: This was done by creating events on the Windows 10 system that would be analysed by the system. The sampling data for testing was identified from literature review as the security and audit events to be monitored. The test data was generated from the Windows 10 system as follows. A user logon event, a failed authentication event by having wrong credentials as inputs to a login, creating a new user, resetting the password of the created user, modification of rights of the created user and deletion of the new user account.

CHAPTER 4: SYSTEM ARCHITECTURE AND DESIGN

4.1 Introduction

This chapter describes the architecture and design of the windows srcurity and event analysis system that satisfies the requirements discovered during the prototyping phase.

4.2 System Architecture

The system architecture of the proposed system can be represented in three tiers.
These are the presentation layer, the data processing layer, and the event storage layer. The presentation tier presents data to the user is web based an to be accessed from a browser. It presents a view processed data and allow users to visualize the processed data into graphs and charts. The data processing layer is used to process events collected by the windows agents from raw unformatted data to indexed information. The event storage layer consists of a denormalized database to store the data indexed by data processing layer.
The architecture include agents (nxlogs) which are installed in the Windows servers, that are used to capture security events and send to the event processor. The event processor (Logstash) converts the received events into indexed data that can be stored in the database (Elasticsearch). When a user wants to view the events they will connect to the visualizing service (Kibana) which pulls data from the database and display to the user.

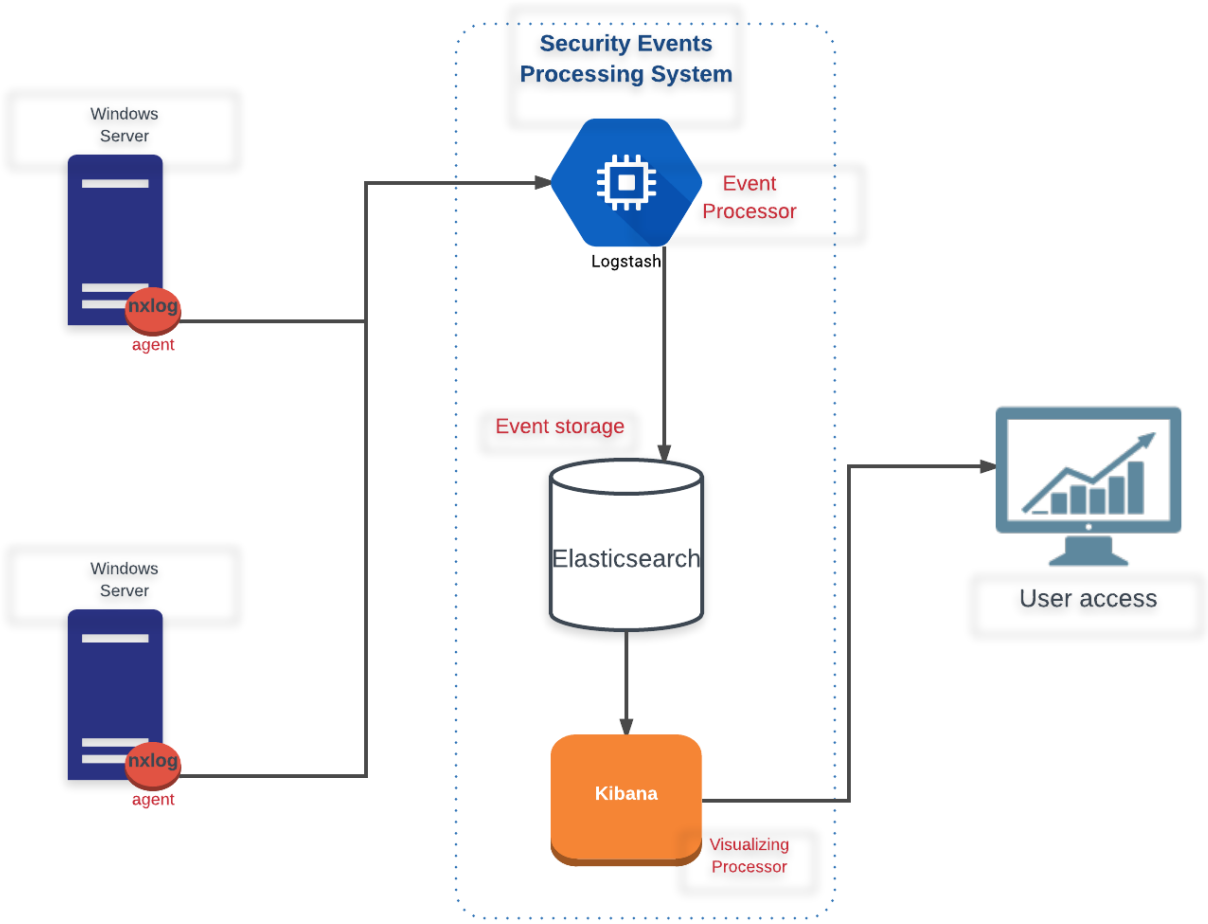

Figure 4.1 Architecture Design

4.3 Sequence Diagram

Figure 4.2 shows a sequence diagram with the processes that occur from the agent up to when the user makes a request.

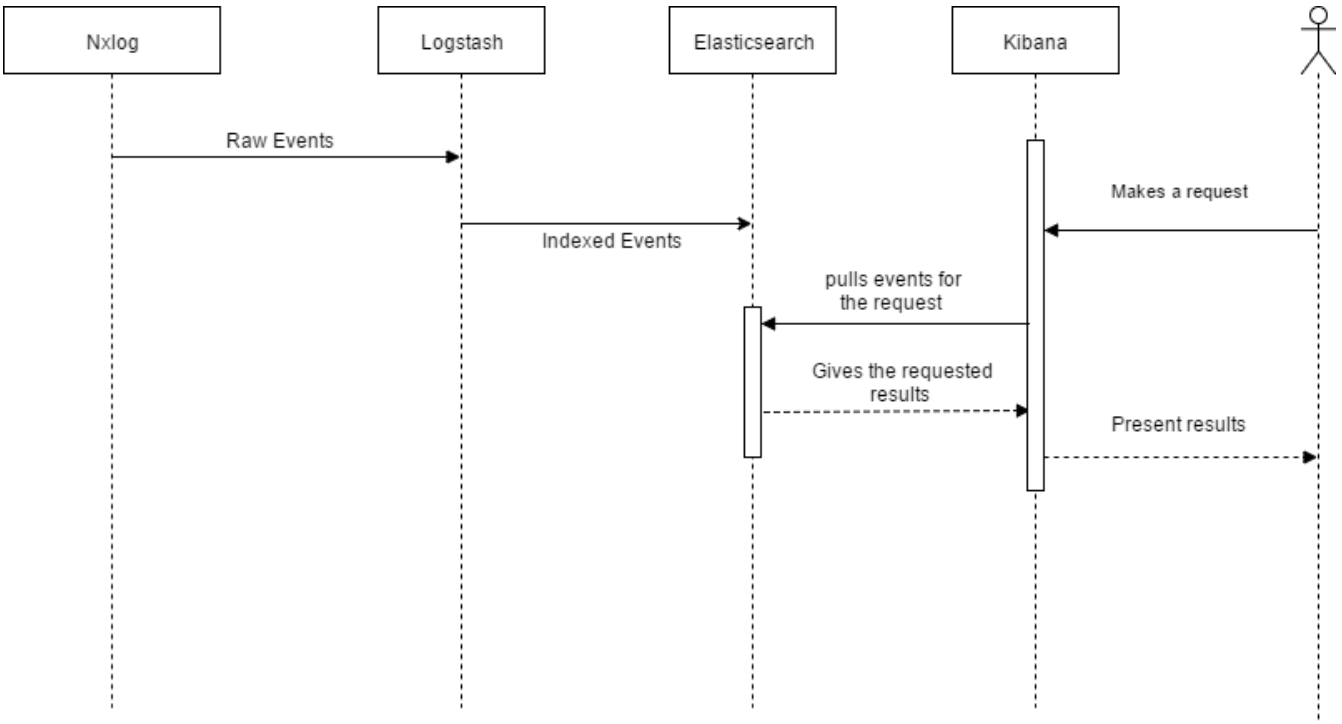

Figure 4.2 Sequence Diagram

4.4 Use Case Modelling

The top level use case diagram of the system is shown in figure 4.2. The only actor is the security analyst who will be the user of the system.

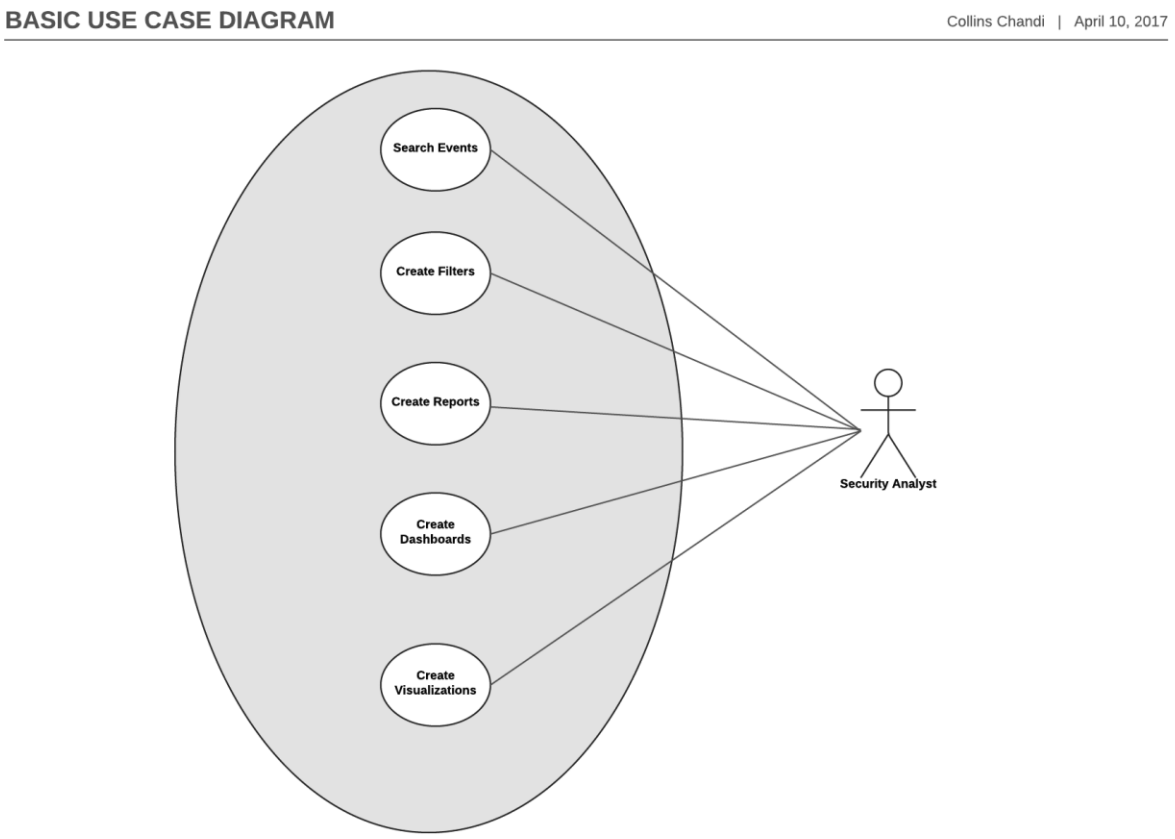

Figure 4.3 Use Case Diagram for security analysis platform

4.4.1 Use Case 1: Search Events

This use case shows how the security analyst will search the system to view security events collected that could be of interest to the organization.

Table 4.1 Search Events Use Case

Use Case Name:	Search Events
Scenario	Done via a web browser
Brief Description	The user will create a search criteria depending on time and

	or specific search strings such as IP Address, username, eventid or any string that could be used to identify an event.	
Actors	Security Analyst	
Preconditions	The system must have received events that can fulfill the search criteria or the will be null results. The events must be processed and indexed appropriately to allow for effective searching of events.	
Post-conditions	Results of the search should be displayed to the user through the web browser.	
Flow of Events (Steps)	**Actor**	**System**
	The user gives an input for a search criteria.	System displays events that qualify to fit under the search criteria.

4.4.2 Use Case 2: Create filters

This use case outlines how the user can create filters, the filters are useful for easy reuse of most common search criterias.

Table 4.2 Create Filters Use Case

Use Case Name:	Create Filters	
Scenario	Done via a web browser	
Brief Description	The user will create a search criteria and save it as a filter.	
Actors	Security Analyst	
Preconditions	The user must create a valid search criteria	
Flow of Events (Steps)	**Actor**	**System**

	The user gives an input for a search criteria and stores it as a filter.	System should store the filter for easy access for the user.

4.4.3 Use Case 3: Create Reports

This use case outlines how the user can create reports, which are in pdf version.

Table 4.3 Create Reports Use Case

Use Case Name:	Create Reports	
Scenario	Done via a web browser	
Brief Description	The user will run a search and export the results as a report	
Actors	Security Analyst	
Preconditions	The user must create a valid search criteria	
Flow of Events (Steps)	**Actor**	**System**
	The user gives an input for a search criteria and exports it as a filter.	System should process the report for the user.

4.4.4 Use Case 5: Create Visualisation

This use case outlines how the user can create visualisations.

Table 4.4 Create Visualisation Use Case

Use Case Name:	Create Visualization
Scenario	Done via a web browser
Brief Description	The user will create graphs and charts based on filters of

	required filters and search queries.
Actors	Security Analyst
Preconditions	The user must have valid search queries and filters.

4.4.5 Use Case 6: Create Dashboards

This use case outlines how the user can create dashboards.

Table 4.5 Create Dasboards Use Case

Use Case Name:	Create Dashboards
Scenario	Done via a web browser
Brief Description	The user will add a collection of related visualizations to an empty dashboard.
Actors	Security Analyst
Preconditions	The user must have created visualizations

4.5 Context Diagram

A context diagram is a component of functional modelling that stands out on its own as a valuable tool (Burge, 2011). Figure 4.4 below represents the context diagram of the system.

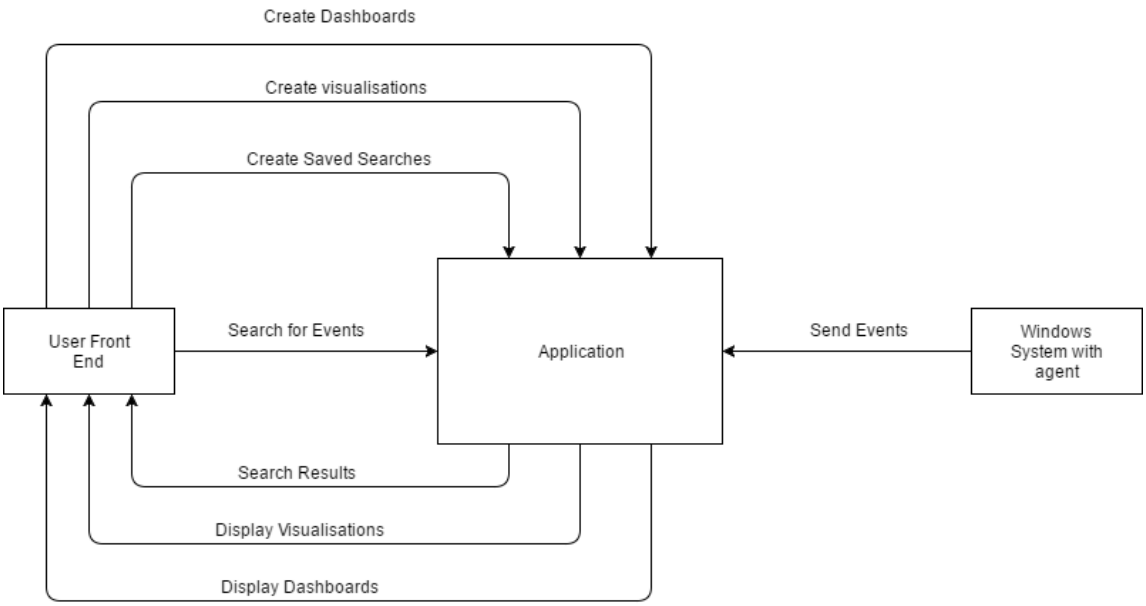

Figure 4.4 Context Diagram

4.6 Database Indexing

The indexing of data coming into elasticsearch is processed into the following categories by kibana per field:

i Type: Where the field is identified either as a string, integer or any other type.

ii Searchable: Some fields are indexed to be searchable through the kibana, meaning that they can be used in the filter bar.

iii Aggregatable: These fields can be used to create aggregates, which is useful in the development of visualization.

iv Analyzed: These are fields that have a value that carries more information and will require more resources e.g. memory to visualize.

Figure 4.5 below gives the index structure as seen through kibana.

name	type	format	searchable	aggregatable	analyzed
SubjectUserSid.keyword	string		✔	✔	
ProcessName	string		✔		✔
LmPackageName.keyword	string		✔	✔	
LogonGuid	string		✔		✔
LogonType.keyword	string		✔	✔	
Category.keyword	string		✔	✔	
TransmittedServices.keyword	string		✔	✔	
type	string		✔		✔
TargetUserName.keyword	string		✔	✔	
HandleId.keyword	string		✔	✔	
ObjectServer	string		✔		✔
Hostname.keyword	string		✔	✔	
Severity.keyword	string		✔	✔	
WorkstationName.keyword	string		✔	✔	
NewSd.keyword	string		✔	✔	
SubjectUserName.keyword	string		✔	✔	
KeyLength	string		✔		✔
tags	string		✔		✔
OldSd	string		✔		✔
EventTime	string		✔		✔
EventID	number		✔	✔	
_source	_source				
RecordNumber	number		✔	✔	
SubjectUserSid	string		✔		✔
ProcessName.keyword	string		✔	✔	
TargetUserSid	string		✔		✔
tags.keyword	string		✔	✔	
Opcode	string		✔		✔
TransmittedServices	string		✔		✔
ObjectType	string		✔		✔
Opcode.keyword	string		✔	✔	
WorkstationName	string		✔		✔

Figure 4.5 Database indexing in Elasticsearch

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

5.1 Introduction

This chapter describes the implementation of the proposed system and highlights the key features. The features and functionality incorporated are the result of incorporating user views and requirements obtained from the evolutionary prototyping phase. This section also includes screenshots of the various end user and back end screens.

This section of this chapter describes the various tests that were carried out on the system. It includes a discussion on the testing processes, locations, and a population that was taken into consideration.

5.2 System Specification

The system holding the applications will have the following specifications all under a virtualized environment:

1. Analysis and event processing system:
    i. Ubuntu server 16.04
    ii. 4 GB RAM
    iii. 20 GB Disk
    iv. Logstash version 5.3.0
    v. Elasticsearch version 5.3.0
    vi. Kibana version 5.3.0
2. Windows 10 Operating System to be tested:
    i. 2 GB RAM
    ii. 20 GB Disk
    iii. Nxlog community edition version 2.9.1716

5.3 System Implementation and Testing

5.3.1 Event Collection Configuration on Windows

The Windows 10 virtual machine was configured to capture audit events by first enabling security events to be audited in the Local Security Policy tool in the Windows machine. This was done to allow the Windows system to generate events when specific actions were taken by a user. The actions were such account logon, account management, policy changes and system events as seen in Figure 5.1.

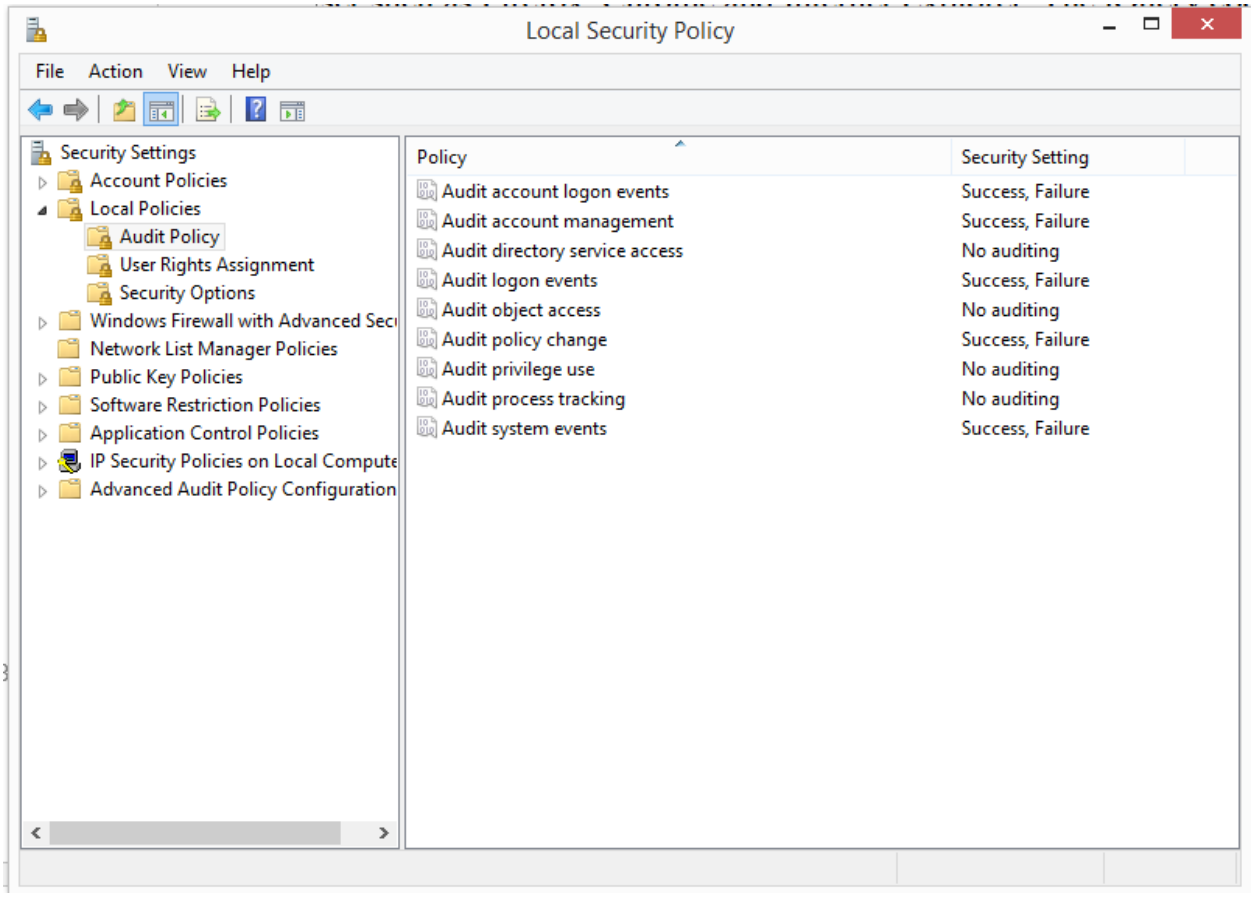

Figure 5.1 Local Security Policy

Installation and configuration of Nxlog agent in the Windows system to capture the audited events. Nxlogs was configured to collect only security events from windows, convert the events into json format and send the events to the Event Analysis Server on

TCP port 5000, see Appendix A. The JSON output sample of what nxlog sends to Logstash can be seen in figure 5.2.

```
2017-04-20 11:12:55 ghost AUDIT_SUCCESS 4670 Permissions on an object were changed.

Subject:
        Security ID:            S-1-5-21-3646898595-1133967514-1974176161-1001
        Account Name:           ghost
        Account Domain:         ghost
        Logon ID:               0x9F962

Object:
        Object Server:  Security
        Object Type:    Token
        Object Name:    -
        Handle ID:      0x2de8

Process:
        Process ID:     0x11f4
        Process Name:   C:\Program Files (x86)\Google\Chrome\Application\chrome.exe

Permissions Change:
        Original Security Descriptor:   D:(A;;GA;;;S-1-5-21-3646898595-1133967514-1974176161-1001)(A;;GA;;;SY)(
        New Security Descriptor:        D:(A;;GA;;;S-1-5-21-3646898595-1133967514-1974176161-1001)(A;;GA;;;SY)
```

Figure 5.2 Nxlog Text Output Sample

5.3.2 Event Analysis Configuration

The individual componenets of the event analysis were installed in Ubuntu 16.04 operating system, see Appendix C. Since each componenet is an independent component the configurations were made to allow for communication between them to be possible through specific ports:

Elasticsearch was configured to be accessed through port 9200.

Kibana was configured to be accessed through port 5601.

Logstash was configured to receive events from nxlog agent using port 5000.

5.3.2.1 Logstash Configuration

The configuration for the Logstash service was made by having the input data source being any event that is captured from port 5000, which is where nxlog is sending data to. The data captured with this input is tagged as tcpjson and given the type nxlog as it is

the source. As source code sample below shows.

```
input {
  tcp {
    codec => json_lines { charset => "UTF-8" }
    port => 5000
    tags => [ "tcpjson" ]
    type => "nxlog"
  }
}
```

The filter for the configuration is a json parser that will capture all data that is of type nxlog and check for those that have the SourceModuleName as eventlog and a mutation function of logstash will replace and remove the Message field. Then capture the EventTime field to be used as the time stamp for the field.

```
filter {
  if [type] == "nxlog" {
    json {
      source => "message"
    }
    if [SourceModuleName] == "eventlog" {
      mutate {
        replace => [ "message", "%{Message}" ]
      }
      mutate {
        remove_field => [ "Message" ]
      }
    }
```

```
  date {
    locale => "en"
    match => [ "EventTime", "YYYY-MM-dd HH:mm:ss" ]
  }
 }
}
```

After the filter has been used to capture the events coming in from the input, processed data can be directed to the output configuration. The output location for this system is the elasticsearch service, which is receiving events from logstash on port 9200.

```
output {
 elasticsearch {
  hosts => [ "localhost:9200" ]
 }
}
```

See Appendix B for full configuration.

To start the Logstash service one has to run the command **service logstash start** if there are no errors given back the service will start.

5.4 System Features

After the installation and configuration of all components both on Windows 10 and Ubuntu Server virtual machines the following were the features available on the security monitoring. platform

5.4.1 Event Searching

When the user accesses the system from the browser they should be able to get the Kibana web user interface where they can search for specific events using strings. Figure 5.3 shows an example of search query made for user "**test**" in the Windows 10 VM that was

created then deleted by the system administrator.

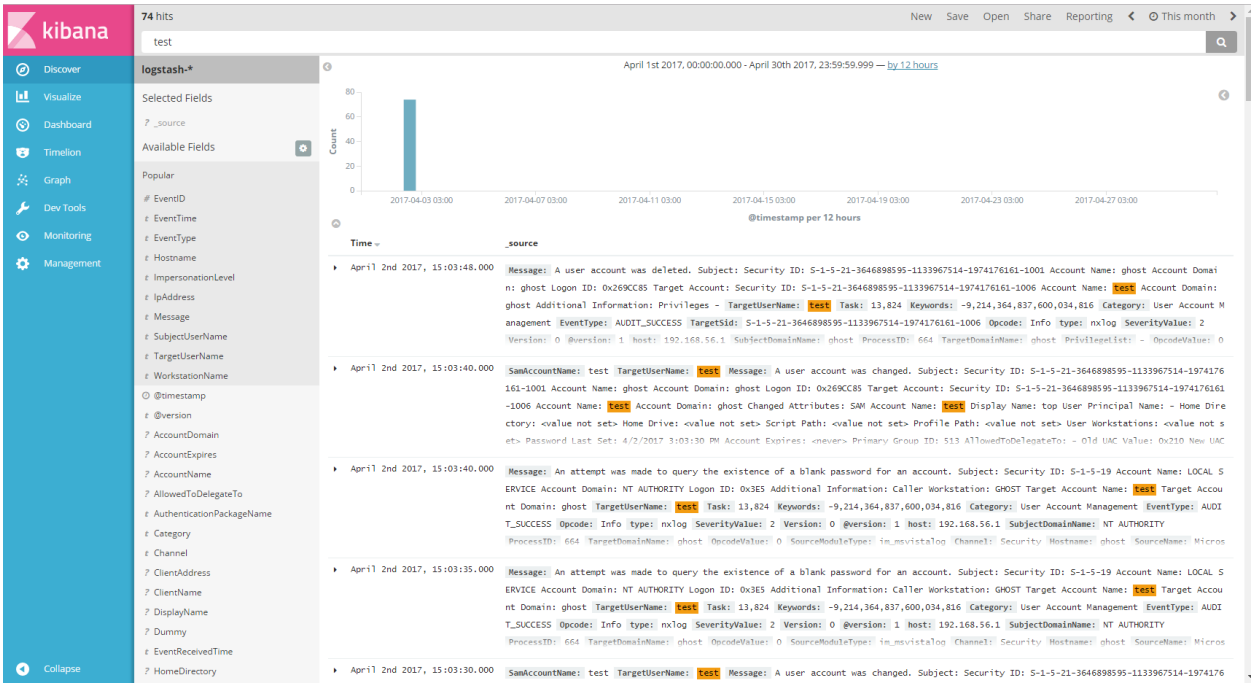

Figure 5.3 Event Searching

5.4.2 *Event Parsing*

Event parsing is done by the Logstash service (in the backend) by analysing the raw events in JSON format and then normalizing and indexing them into specific fields. Figure 5.4 shows the raw log for an event relating to the deletion of user "**test**".

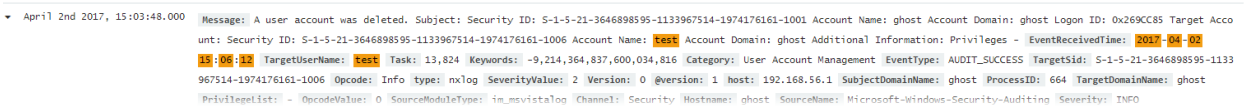

Figure 5.4 Event Parsing Raw Log

When normalized and indexed the raw log in Figure 5.4 can be expanded into a more detailed output with specified fields giving information on each part of the event log as seen in Figure 5.5.

@timestamp		April 2nd 2017, 15:03:48.000
@version		1
Category		User Account Management
Channel		Security
EventID		4,726
EventReceivedTime		2017-04-02 15:06:12
EventTime		2017-04-02 15:03:48
EventType		AUDIT_SUCCESS
Hostname		ghost
Keywords		-9,214,364,837,600,034,816
Message		A user account was deleted. Subject: Security ID: S-1-5-21-3646898595-1133967514-1974176161-1001 Account Name: ghost Account Domain: ghost Logon ID: 0x269CC85 Target Account: Security ID: S-1-5-21-3646898595-1133967514-1974176161-1006 Account Name: test Account Domain: ghost Additional Information: Privileges -
Opcode		Info
OpcodeValue		0
PrivilegeList		-
ProcessID		664
ProviderGuid		{54849625-5478-4994-A5BA-3E3B0328C30D}
RecordNumber		185,944
Severity		INFO
SeverityValue		2
SourceModuleName		in
SourceModuleType		im_msvistalog
SourceName		Microsoft-Windows-Security-Auditing
SubjectDomainName		ghost
SubjectLogonId		0x269cc85
SubjectUserName		ghost
SubjectUserSid		S-1-5-21-3646898595-1133967514-1974176161-1001
TargetDomainName		ghost
TargetSid		S-1-5-21-3646898595-1133967514-1974176161-1006
TargetUserName		test
Task		13,824
ThreadID		6,656
Version		0
_id		AVsujv8KzzPBHA9UwZvb
_index		logstash-2017.04.02

Figure 5.5 Event Parsing Normalized

5.4.3 *Creating a Saved Search*

A saved search is a search query that is saved on the Kibana for reuse in the future. This is mainly for common search parameters that the user may occasionally use. The saved searches are commonly used as a pre-requisite when one wants to create some visualisation. Figure 5.6 shows a list of saved searches that the user may reuse for Windows Security event analysis.

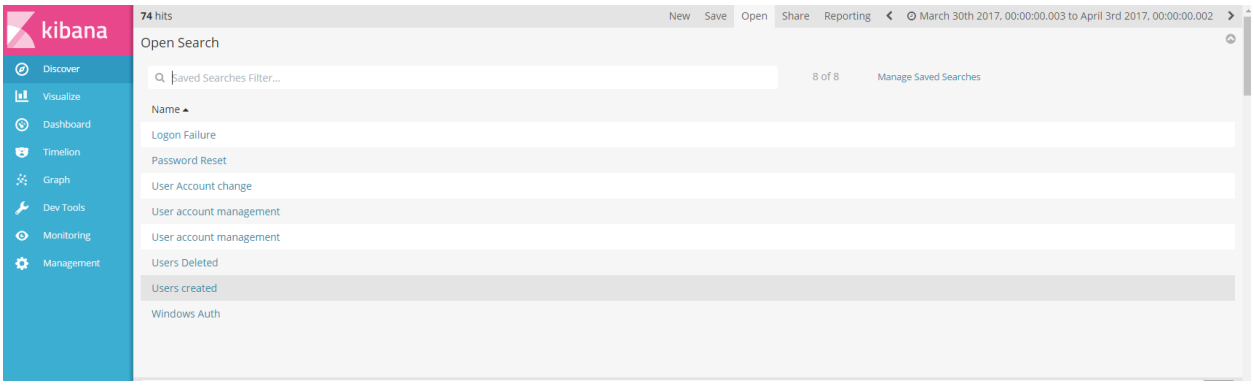

Figure 5.6 Saved Search

After running a successful search for events on Kibana, there is an option that allows one to save the search, figure 5.7 shows an example.

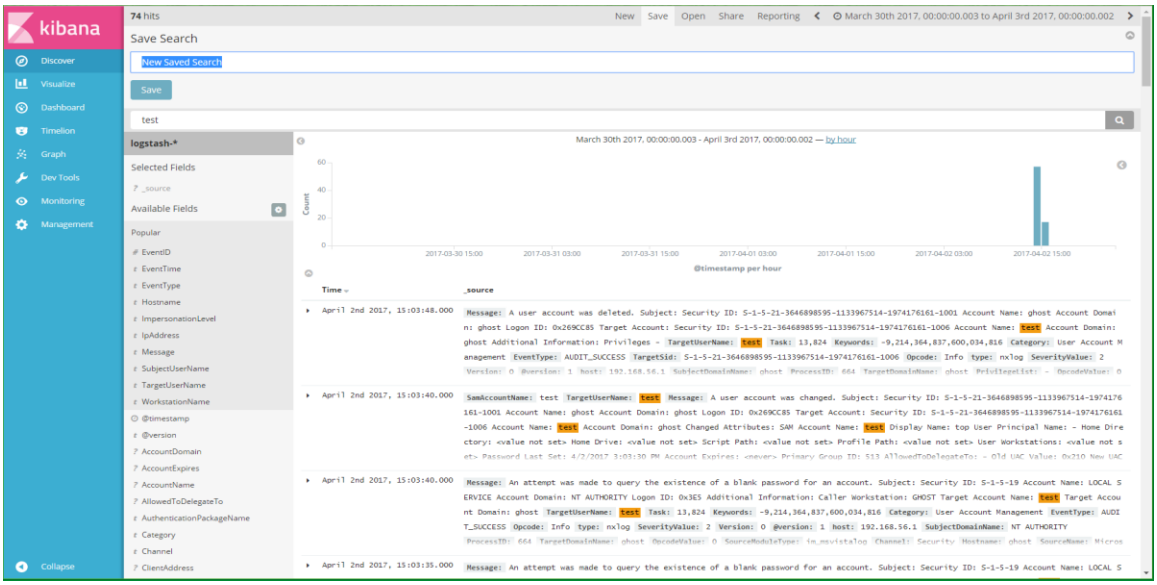

Figure 5.7 Saving a Search

5.4.4 *Creating a Visualisation*

Visualisations will aid the security analyst in getting high level over view of events collected by the system. These are created from the saved searches and will used the indexed fields in normalised events as options when it comes to creating charts and graphs. Figure 5.8 shows an example of a visualisation of the top five windows Event IDs with specific users captured by the system.

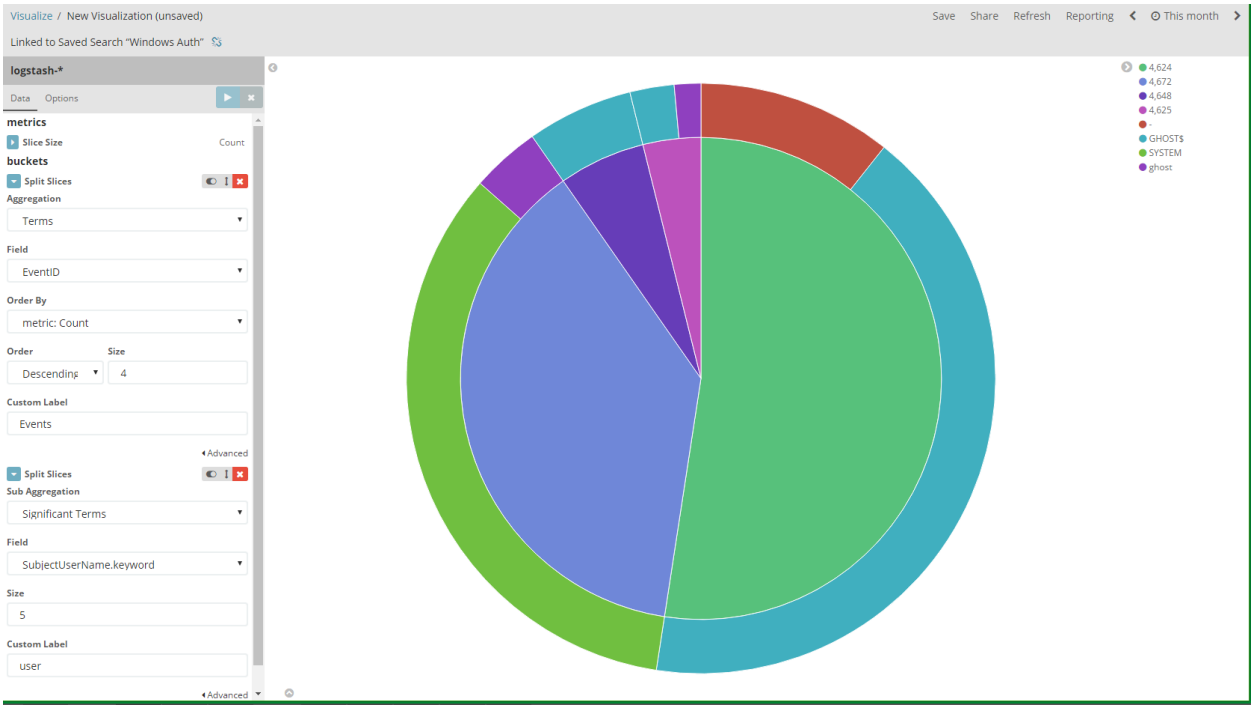

Figure 5.8 Visualisation

5.4.5 Creating Reports

With saved searches and visualisations being available and having relevant data in them one can generate reports form them. The reporting option in Kibana converts the results given in HTML via the web browser into PDF files which the user can download and share. Kibana provides a Reporting manager tool that will handle the conversion of HTML to PDF as seen through figures 5.9 to 5.11

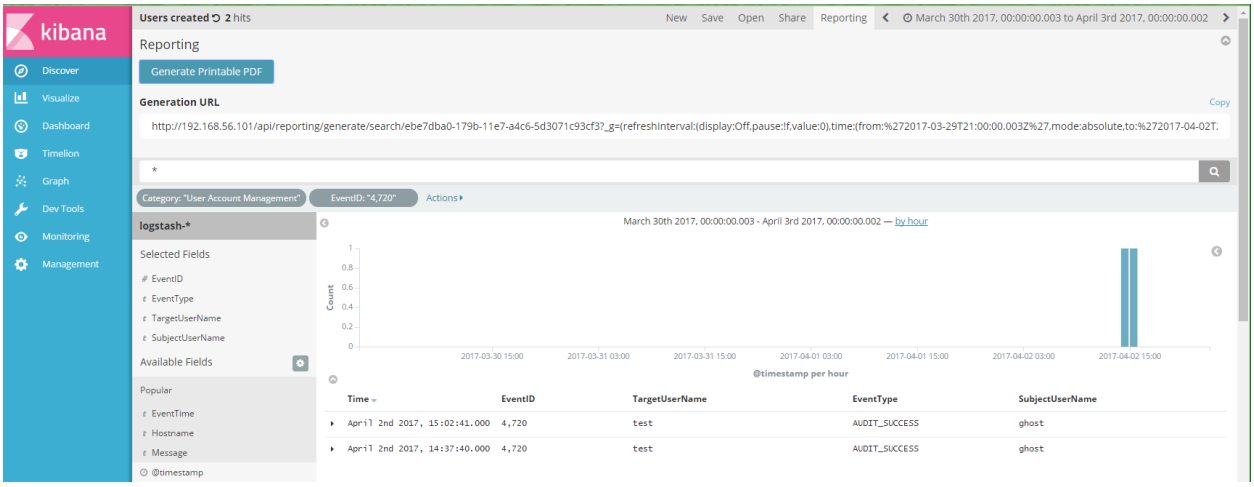

Figure 5.9 Saved Search Used For Reporting

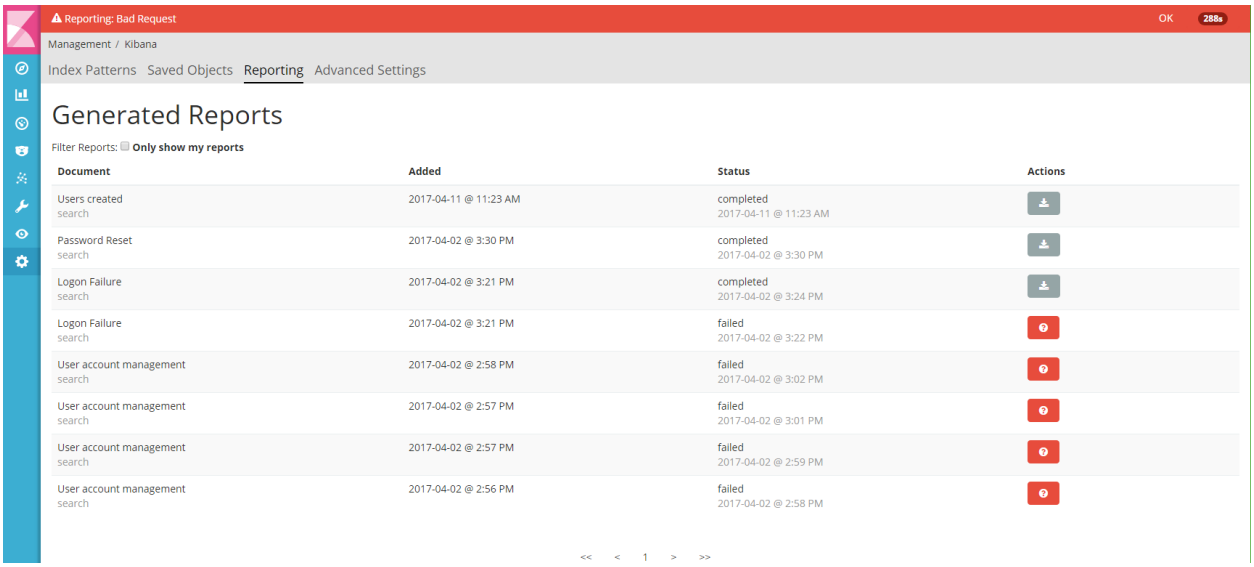

Figure 5.10 Reporting Manager

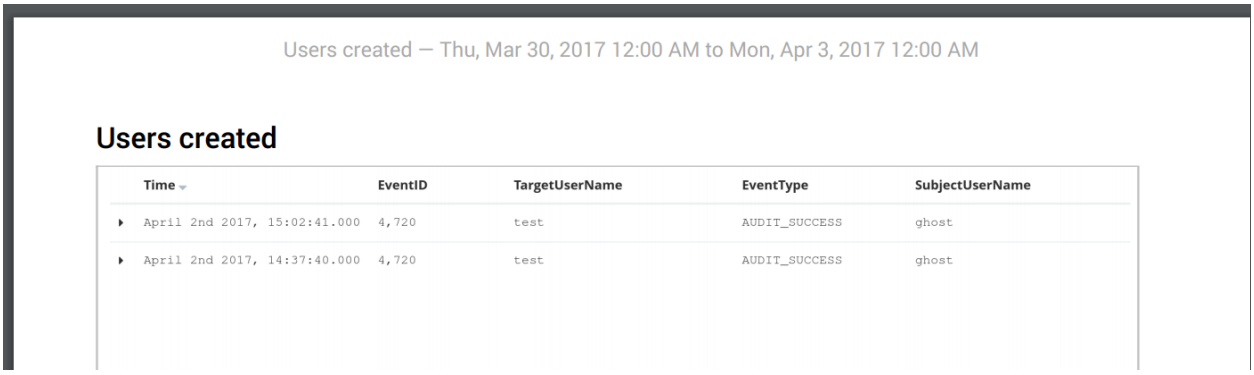

Figure 5.11 Report Output

5.4.6 Creating Dashboards

Dashboards are a collection of several Visualisations as set by the system user depending on points of interest.

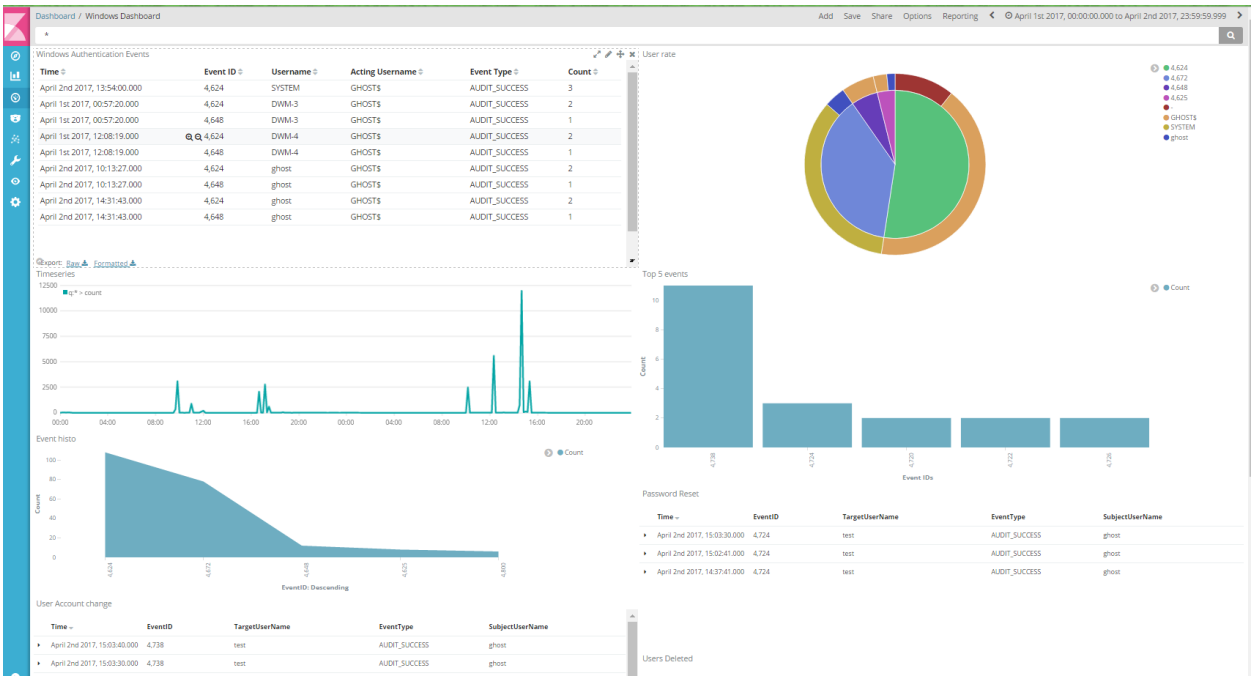

Figure 5.12 Dashboard

5.5 System Testing

This Section covers testing of the system to ensure that it works well. The testing was divided into two sections, developer testing and user Assessment testing. The first tests done by the developer were to ensure that the various functionalities were working well, the tests included: unit testing, compatibility testing, functionality testing.

5.5.1 Functionality testing

Testing was done by creating events on the Windows 10 system that were captured by the analysis platform. The events were:

i. User Login with user ghost.
ii. Failed authentication event with user ghost to simulate a bruteforce attacke by checking the count of failed authentication attacks agains one user account.
iii. Creation of user test by user ghost
iv. Password reset for user test by user ghost.
v. User Account modification of user test.
vi. Deletion of user test by user ghost.

The results for the tests can be seen in Figure 5.13 where a dashboard was created to capture these events after analysis and grouped to different event tables.

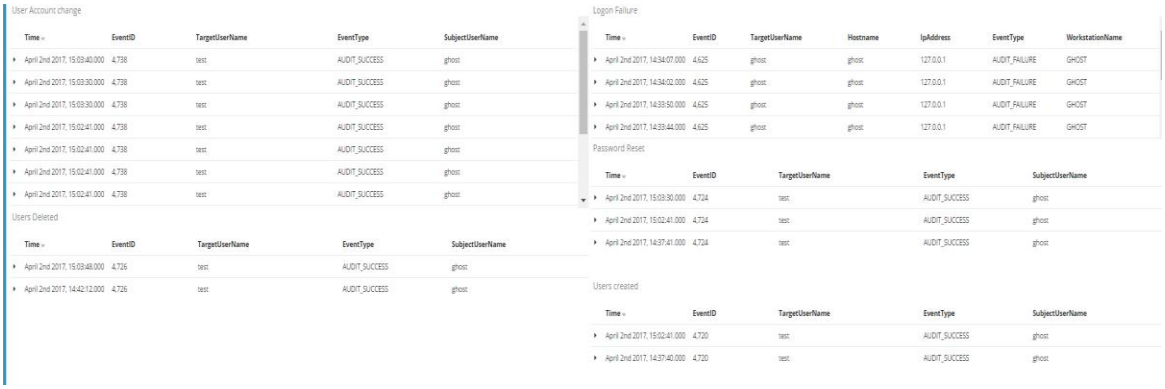

Figure 5.13 Test Results on Dashboard

Unit Testing was done after configuration of different components to ensure that there

were no failures on them.

5.5.2 Unit and Intergration Testing

In unit testing, the individual units were scrutinized to test for operation. The isolated software components were separately tested to ensure that each component performs required function. Integration testing was performed after system integration where two or more of the components were integrated and tested for their functionality. During this test, some errors were deduced and corrected. The test were done by checking if the individual componenets of the system were running by checking if their service ports were open.

5.5.2.1 Logstash

The Logstash service was set to receive events using TCP port 5000. The test was to check if the port was open and if it was receiving any data from the agent. This was done by running tcpdump command to check if the port was receiving any data.

```
root@ubuntu:~# tcpdump -i enp0s3 -vn port 5000
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
12:46:16.960274 IP (tos 0x0, ttl 128, id 26403, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.56.1.52142 > 192.168.56.101.5000: Flags [F.], cksum 0x92eb (correct), seq 1975897966, ack 3765577043, win 256, length 0
12:46:16.990518 IP (tos 0x0, ttl 64, id 23658, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.56.101.5000 > 192.168.56.1.52142: Flags [F.], cksum 0xf1d1 (incorrect -> 0x82a7), seq 1, ack 1, win 4419, length 0
12:46:16.990673 IP (tos 0x0, ttl 128, id 26404, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.56.1.52142 > 192.168.56.101.5000: Flags [.], cksum 0x92ea (correct), ack 2, win 256, length 0
```

Figure 5.14 Logstash Tcpdump Output

Figure 5.14 shows us that the port was receiving data as expected hence the service was running properly.

```
root@ubuntu:~# lsof -i :5000 -S
COMMAND PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java    991 logstash   47u  IPv6  18890      0t0  TCP *:5000 (LISTEN)
java    991 logstash   62u  IPv6  55203      0t0  TCP 192.168.56.101:5000->192.168.56.1:53511 (ESTABLISHED)
java    991 logstash   69u  IPv6  61644      0t0  TCP 192.168.56.101:5000->192.168.56.1:57877 (ESTABLISHED)
java    991 logstash   73u  IPv6  62437      0t0  TCP 192.168.56.101:5000->192.168.56.1:52142 (ESTABLISHED)
root@ubuntu:~#
```

Figure 5.15 Logstash Connections

Figure 5.15 shows the connections that are active with the Logstash application, here we

can see the IP 192.168.56.1 is connected to the system's IP 192.168.56.101 on the port 5000.

5.5.2.2 Elasticsearch

The Elasticsearch service was set to receive data from Logstash on port 9200 and also receive requests from Kibana on the port 9200. The figure 5.16 below shows the connections that were active when Elasticsearch was connected to both Kibana and Logstash.

```
root@ubuntu:~# lsof -i :9200 -S
COMMAND  PID          USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java     991      logstash   18u  IPv6  18320      0t0  TCP localhost:50130->localhost:9200 (ESTABLISHED)
java     991      logstash   40u  IPv6  15241      0t0  TCP localhost:50132->localhost:9200 (ESTABLISHED)
java     991      logstash   66u  IPv6  55207      0t0  TCP localhost:52412->localhost:9200 (ESTABLISHED)
java     991      logstash   67u  IPv6  55208      0t0  TCP localhost:52414->localhost:9200 (ESTABLISHED)
java     991      logstash   68u  IPv6  59208      0t0  TCP localhost:52416->localhost:9200 (ESTABLISHED)
java    1062 elasticsearch  195u  IPv6  18213      0t0  TCP localhost:9200 (LISTEN)
java    1062 elasticsearch  196u  IPv6  18214      0t0  TCP localhost:9200 (LISTEN)
java    1062 elasticsearch  285u  IPv6  51128      0t0  TCP localhost:9200->localhost:51602 (ESTABLISHED)
java    1062 elasticsearch  290u  IPv6  33551      0t0  TCP localhost:9200->localhost:50178 (ESTABLISHED)
java    1062 elasticsearch  294u  IPv6  15242      0t0  TCP localhost:9200->localhost:50132 (ESTABLISHED)
java    1062 elasticsearch  296u  IPv6  33552      0t0  TCP localhost:9200->localhost:50180 (ESTABLISHED)
java    1062 elasticsearch  297u  IPv6  15240      0t0  TCP localhost:9200->localhost:50130 (ESTABLISHED)
java    1062 elasticsearch  298u  IPv6  33553      0t0  TCP localhost:9200->localhost:50182 (ESTABLISHED)
java    1062 elasticsearch  299u  IPv6  33554      0t0  TCP localhost:9200->localhost:50184 (ESTABLISHED)
java    1062 elasticsearch  300u  IPv6  33559      0t0  TCP localhost:9200->localhost:50188 (ESTABLISHED)
java    1062 elasticsearch  303u  IPv6  33565      0t0  TCP localhost:9200->localhost:50196 (ESTABLISHED)
java    1062 elasticsearch  304u  IPv6  33555      0t0  TCP localhost:9200->localhost:50186 (ESTABLISHED)
java    1062 elasticsearch  306u  IPv6  30377      0t0  TCP localhost:9200->localhost:50394 (ESTABLISHED)
java    1062 elasticsearch  308u  IPv6  30378      0t0  TCP localhost:9200->localhost:50396 (ESTABLISHED)
java    1062 elasticsearch  309u  IPv6  30379      0t0  TCP localhost:9200->localhost:50398 (ESTABLISHED)
java    1062 elasticsearch  313u  IPv6  51148      0t0  TCP localhost:9200->localhost:51628 (ESTABLISHED)
java    1062 elasticsearch  320u  IPv6  56328      0t0  TCP localhost:9200->localhost:51634 (ESTABLISHED)
java    1062 elasticsearch  328u  IPv6  57886      0t0  TCP localhost:9200->localhost:52412 (ESTABLISHED)
java    1062 elasticsearch  329u  IPv6  57887      0t0  TCP localhost:9200->localhost:52414 (ESTABLISHED)
java    1062 elasticsearch  338u  IPv6  59209      0t0  TCP localhost:9200->localhost:52416 (ESTABLISHED)
node    2205        kibana   10u  IPv4  46881      0t0  TCP localhost:50178->localhost:9200 (ESTABLISHED)
node    2205        kibana   11u  IPv4  46883      0t0  TCP localhost:50180->localhost:9200 (ESTABLISHED)
node    2205        kibana   12u  IPv4  30180      0t0  TCP localhost:50182->localhost:9200 (ESTABLISHED)
node    2205        kibana   13u  IPv4  46897      0t0  TCP localhost:50186->localhost:9200 (ESTABLISHED)
node    2205        kibana   14u  IPv4  30181      0t0  TCP localhost:50184->localhost:9200 (ESTABLISHED)
node    2205        kibana   16u  IPv4  48739      0t0  TCP localhost:50188->localhost:9200 (ESTABLISHED)
node    2205        kibana   18u  IPv4  30187      0t0  TCP localhost:50196->localhost:9200 (ESTABLISHED)
node    2205        kibana   19u  IPv4  56244      0t0  TCP localhost:51602->localhost:9200 (ESTABLISHED)
node    2205        kibana   20u  IPv4  51217      0t0  TCP localhost:50394->localhost:9200 (ESTABLISHED)
node    2205        kibana   21u  IPv4  51218      0t0  TCP localhost:50396->localhost:9200 (ESTABLISHED)
node    2205        kibana   22u  IPv4  51219      0t0  TCP localhost:50398->localhost:9200 (ESTABLISHED)
node    2205        kibana   25u  IPv4  56321      0t0  TCP localhost:51628->localhost:9200 (ESTABLISHED)
node    2205        kibana   27u  IPv4  56327      0t0  TCP localhost:51634->localhost:9200 (ESTABLISHED)
```

Figure 5.16 Elasticsearch Connections

5.5.2.3 Kibana

Kibana is a service that runs on the port 5601, and it was running by listening for connections using the port actively as figure 5.17 shows.

```
root@ubuntu:~# lsof -i :5601 -S
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
node    2205 kibana   15u  IPv4  48737      0t0  TCP localhost:5601 (LISTEN)
root@ubuntu:~#
```

Figure 5.17 Kibana Service

5.5.3 Compatibility Testing

Compatibility testing was done to ensure that the web application is compatible with the available platforms. As table 5.1 below shows.

Table 5.1 Browser Compatibility Tests

Browser	Compatibility
Firefox (version 8.0 and above)	Yes
Internet Explorer (versions 4 and above)	Yes
Chrome (versions 40 and above)	Yes

CHAPTER 6: DISCUSSIONS

6.1 Introduction

The purpose of the dissertation was to review the existing solutions used in threat monitoring and event collection, to investigate the current techniques used for event analysis, to design and develop and test a platform that can collect, analyse and visualise Windows security events and to validate the effectiveness of the proposed solution. This was done in order to identify and develop a suitable technique that will be used by security analysts to monitor the security of windows systems in an organization. The research findings helped in identifying the appropriate technique which was adopted and a platform for collecting and analysing windows security and audit events then visualising them for the users ease.

6.2 Findings and Achievements

A review of the literature indicated the techniques most organizations cannot detect security breaches in their systems due to lack of effective monitoring. Security intrusions that occur can run up to 80 days without the security team knowing about it while containment of the security breach is done after 2 days. This shows that if the process of detecting security breaches can be improved, then an organization can contain the any security breach a few days there after. The cost of impelementing a proprietary security monitoring tool is a challenge to most organizations.

The developed system is an effective way to ensure that security breaches that occour on windows systems are identified as early as possible. The developed system is based on open-source tools that are widely documented and supported meaning that the cost is not a major factor when choosing this system. This project will therefor improve the security monitoring of organizations that ran on windows systems for their services.

6.3 Review of Research Objectives

This dissertation identifies the challenges faced by organizations in the identifying security breaches in windows systems based on journals, websites and books. A security monitoring system was designed and develop with a selected technique from the literature review and results from system analysis. The research objectives acted as a guideline to develop the system.

The first objective was to identify the purpose of security monitoring and event collection, this was achieved using litereature review. Most organizations fail to detect intrusions due to lack of proper monitoring of security events generated by the IT infrastructure. The collection, analysis and monitoring of security events was also noted to be part of best practice that is manadatory for organizations that seek to attain certain compliance certifications such as HIPAA and SOX. This shows that security events monitoring is key to ensuring that an organization is up to par with best practices while ensuring that they are able to secure the organization with early detection of incidents.

The second objective was to review the existing solutions used in threat monitoring and event collection. This was achieved through reviewing secondary data and reading journals article. The review of the secondary data revealed that the existing solutions require the organizations to have competent security personel to use them in order to monitor security. The analysis systems used relied on tools such as regular expressions and there were limited options when it came to visualising of security events. The cost of running security management systems is also too high for organizations and many end up relying on vendors for deployment and the cost hiring and training of analysts is also factored in.

The third objective was to design and develop and test a platform that can collect, analyse and visualise Windows security events. This objective was achieved through the design, implementation and testing of the system. It was developed using the combination of three core services, namely Logstash, Elasticsearch and Kibana. The following tests were

carried out; functional testing where the system functionality was tested against several security events that were generated from the Windows system, compatibility testing was tested against different web browsers and unit testing was done to check if seperate componenets were working effectively.

The fourth objective was to validate the effectiveness of the proposed solution, this was done by running tests with use cases of what an attacker would do with access to a Windows system. The tests proved that it was possible for an analyst to serach for security events create saved searches for reuse, create visualisations that were used in creating dashboards to display analysed security events. The solution provided can be used during forensic investigations if implemented at a whole scale by showing evidence of user activity in relation to security and auditable events in the workstations and servers being monitored.

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

This dissertation developed a platform for the monitoring of security and audit events for windows systems that can be used by organizations in monitoring security incidents. This comes as an aid for organizations to ensure that they are able to protect their organization's information security assets from malicious actors. The solution was based of tools that are open source and widely contributed to by the community, the only expense and limitation during deployment of the platform is that of hardware resources that an organization may have.

7.2 Recommendations

This dissertation only focused on the Windows operating system as a source of security and audit events but the system is not limited to that. The platform can be configured to collect and analyse security events from Unix based systems that have syslog, networking devices and custom applications that an organization may have. However when deploying one will have to configure analysis rules to allow other sorces of security events.

7.3 Future Works

An expansion into event collection from other sources and intergration with Threat Intelligence feeds will highly improve the knowledge base for security event analysis and monitoring. Artificial Intelligence can also be used by learning an organizations normal security events and alarming when suscpiscious events occur in the organization.

REFERENCES

Alienvault. (2014). *The Top 5 Problems With Traditional SIEM*. Retrieved Feb 25, 2017, from Information Security Buzz: http://www.informationsecuritybuzz.com/articles/top-5-problems-traditional-siem/

Anderson, B., & Rundle, R. (2017). *Connect Windows computers to Azure Log Analytics*. Retrieved Nov 12, 2016, from https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-windows-agents

Apache. (2007, 1 9). *Chainsaw*. Retrieved 6 2, 2017, from Logging Services: http://logging.apache.org/chainsaw/index.html

Banon, S. (2010). *The Future of Compass & Elasticsearch*. Retrieved Feb 12, 2017, from http://thedudeabides.com/articles/the_future_of_compass

Batchellor, V. (2016). *Get the security budget you need and spend it wisely*. Retrieved Dec 08, 2016, from https://securityintelligence.com/get-the-security-budget-you-need-and-spend-it-wisely/

Beal, V. (2011). *Mobile operating systems (mobile OS) explained by Webopedia.com*. Retrieved Dec 11, 2016, from http://www.webopedia.com/DidYouKnow/Hardware_Software/mobile-operating-systems-mobile-os-explained.html

Bryman , J. M., & Bell, E. (2007). Business Research Methods Revised Editon. Oxford University Press.

Burke, J. (n.d.). *Log management and analysis: How, when and why*. Retrieved Dec 13, 2011, from http://searchsecurity.techtarget.com/video/Log-management-and-analysis-How-when-and-why

Cavanagh, L. (2016, Jan 2). *How can we improve Azure Search?* Retrieved May 30, 2017, from Microsoft Azure: https://feedback.azure.com/forums/263029-azure-search/suggestions/6328651-support-log-analytics

centeractive. (2015). *retrospective*. Retrieved 6 2, 2017, from centeractive: http://www.retrospective.centeractive.com/log-analyzer

Chapple, E. (2016). *Announcing the launch of windows server 2016*. Retrieved Nov 10, 2016, from https://blogs.technet.microsoft.com/hybridcloud/2016/09/26/announcing-the-launch-of-windows-server-2016/

Cholakian, A. (2014). *Exploring Elasticsearch*. Retrieved Mar 23, 2017, from http://exploringelasticsearch.com/overview.html#sec-over-what-is-elasticsearch

Cid, D. B. (2007). *Log Analysis using OSSEC.* Retrieved Mar 26, 2017, from Ossec: http://ossec.net/ossec-docs/auscert-2007-dcid.pdf

Cid, D., Hay, A., & Bray, R. (2008). *OSSEC Host-Based Intrusion Detection Guide* (1 ed.). Syngress.

Cordray, R. (2015). *Why log management is absolutely critical for IT security*. Retrieved Nov 25, 2016, from http://www.itworldcanada.com/blog/why-log-management-is-absolutely-critical-for-it-security/377711

Costea, A. (2016). *How to configure Windows Event Log Forwarding*. Retrieved Dec 19, 2016, from http://www.vkernel.ro/blog/how-to-configure-windows-event-log-forwarding

Crosby, W. (2016). *Introduction to secure software development life cycle*. Retrieved Jan 11, 2017, from http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/

Darcy, J. (2014). *Why did Microsoft windows become worldś most popular operating system?* Retrieved Nov 18, 2016, from https://www.quora.com/Why-did-Microsoft-

Windows-become-worlds-most-popular-Operating-System

EY. (2014). *Insights on governance, risk and compliance.* EY. Retrieved Nov 17, 2016, from http://www.ey.com/Publication/vwLUAssets/EY-security-operations-centers-helping-you-get-ahead-of-cybercrime/$FILE/EY-security-operations-centers-helping-you-get-ahead-of-cybercrime.pdf

Fadilpasic, S. (2016, March 16). *SIEM is expensive for 69 per cent of companies*. Retrieved Dec 10, 2016, from IT Pro Portal: http://www.itproportal.com/2016/03/16/siem-is-expensive-for-69-per-cent-of-companies/

Gemino, A., & Parker, D. (2009). *Use case diagrams in support of use case modeling: Deriving understanding from the picture.* Journal of Database Management.

Glover, G. (2015). *The importance of log management*. Retrieved Jan 29, 2017, from http://blog.securitymetrics.com/2015/08/importance-of-log-management.html

HHS. (2015). *HIPAA for professionals*. Retrieved Nov 12, 2016, from https://www.hhs.gov/hipaa/for-professionals/index.html

Hoffman, C. (2012). *What Is the Windows Event Viewer, and How Can I Use It?* Retrieved Mar 22, 2017, from https://www.howtogeek.com/123646/htg-explains-what-the-windows-event-viewer-is-and-how-you-can-use-it/

Hruska, J. (2016). *Windows drops below 90% market share for the first time in years; windows 7 falls below 50%*. Retrieved Nov 12, 2016, from https://www.extremetech.com/computing/227693-windows-drops-below-90-market-share-for-the-first-time-in-years-windows-7-falls-below-50

Ipswitch. (2010). *BEST PRACTICES: EVENT LOG MANAGEMENT FOR SECURITY AND COMPLIANCE INITIATIVES.* Retrieved Nov 20, 2016, from https://www.ipswitch.com/Ipswitch/media/Ipswitch/Documents/Resources/Whitepapers%20and%20eBooks/ELM_Security_WP.pdf?ext=.pdf

Karen Kent, M. P. (2006). *Guide to Computer Security Log Management.* National Institute of Standards & Technology.

Khan, S. (2013, oct 5). *Elasticsearch Basics*. Retrieved 6 4, 2017, from Slideshare: https://www.slideshare.net/shifa27/elasticsearch-26896932?from_action=save

Kothari, C. (2004). *Research Methodology.* New Age International.

Kumar, V. (2012). *Security Information Management vs Security Event management vs Security Information and Event Management*. Retrieved Dec 23, 2016, from https://www.symantec.com/connect/articles/security-information-management-vs-security-event-management-vs-security-information-and-ev

Levy, Y., & Ellis, T. J. (n.d.). *2011.*

Lewis, R. (2014). *5 great reasons to store and analyze centralized logs*. Retrieved Nov 17, 2016, from https://www.auvik.com/media/blog/reasons-centralized-logs/

Magalhaes, R. (2003). *Search*. Retrieved Mar 19, 2017, from http://techgenix.com/understanding_windows_logging/

Manes, C. (2016). *Most vulnerable operating systems and applications in 2015*. Retrieved Nov 12, 2016, from http://techtalk.gfi.com/2015s-mvps-the-most-vulnerable-players/

Microsoft. (2011). *Event types*. Retrieved Dec 13, 2016, from https://msdn.microsoft.com/en-us/library/windows/desktop/aa363662(v=vs.85).aspx

Microsoft. (2016). *Audit policy*. Retrieved Dec 13, 2016, from https://technet.microsoft.com/en-us/library/cc766468(v=ws.10).aspx

Mishra, J., & Mohanty, A. (2012). *Software Engineering.* New Delhi, India: Dorling Kindersley.

Mohapatra, S., & Joseph, P. (2014). *Management Information Systems in the Knowledge Economy.* PHI Learning.

Mullinix, M. (2013). *An Analysis of Microsoft Event Logs.* Utica College. Retrieved from http://programs.online.utica.edu/sites/uti/files/Mullinix_4_Gonnella_An_Analysis_of_Microsoft_Event_Logs_December_2013.pdf

Nanda, N., & Kumar, S. (2001). *Log it or lose it*. Retrieved Nov 13, 2016, from

http://www.javaworld.com/article/2075660/testing-debugging/log-it-or-lose-it.html

Netwrix. (2016). *Event Log Management with Netwrix Auditor*. Retrieved Dec 09, 2016, from Netwrix: https://www.netwrix.com/event_log_retention_requirements_compliance_regulations.html

Parriott, D. (2013). *ossec/ossec-wui*. Retrieved Jan 23, 2017, from https://github.com/ossec/ossec-wui

Raab, H. (2011, Oct 13). *Log Expert*. Retrieved Jun 2, 2017, from Log Expert: http://logexpert.codeplex.com/documentation

RiskFocus. (2014, August). *Splunk vs. ELK: Part 1 Cost*. Retrieved Jan 12, 2017, from Risk Focus: https://riskfocus.com/splunk-vs-elk-part-1-cost/

Rouse, M. (2013). What is operating system (OS)? - definition from WhatIs.com. Retrieved from http://whatis.techtarget.com/definition/operating-system-OS

Rouse, M. (2014). What is Microsoft windows? - definition from WhatIs.com. Retrieved from http://searchwindowsserver.techtarget.com/definition/Windows

Schmidt, W. (2012). *SIM, SEM, SIEM, what does it mean?* Retrieved Nov 24, 2016, from http://www.wernerschmidt.com/blog/index_files/6bf90acaf61fb4b4a614a6453df16439-16.html

Sugano, A. (2003). OS event-log monitoring. Retrieved from http://windowsitpro.com/systems-management/os-event-log-monitoring

Tenable. (2015). Log management & SIEM. Retrieved from https://www.tenable.com/solutions/log-management-siem

Trustwave. (2015). *2015 TRUSTWAVE GLOBAL SECURITY REPORT.* Retrieved Dec 12, 2016, from http://www2.trustwave.com/rs/815-RFM-693/images/2015_TrustwaveGlobalSecurityReport.pdf

Trustwave. (2016). *Trustwave global security report 2016.* Retrieved Nov 13, 2016, from

https://www.info-point-security.com/sites/default/files/open_downloads/Trustwave_Global_Security_Report_2016.pdf

Tubin, G. (2013). *Endpoint security: No Admin rights, no Malware? Yeah, right!* Retrieved Dec 23, 2016, from https://securityintelligence.com/endpoint-security-admin-rights-malware-yeah-right/

Warda, S. (2017, Feb 12). *Choosing centralized logging and monitoring system*. Retrieved Jun 3, 2017, from IndexOutOfRange: https://indexoutofrange.com/Choosing-centralized-logging-and-monitoring-system/

William, S. (2000). *Microsoft Windows 2000 Administratorś Pocket Consultant.* Microsoft Press.

Zawinski, J. (2011). *The Problem With Jamie Zawinski and Regular Expressions Coyote Tracks*. Retrieved Apr 7, 2017, from http://kagan.mactane.org/blog/2011/08/16/the-problem-with-jamie-zawinski-and-regular-expressions/

APPENDICES

Appendix A: Nxlog Configuration

```
## This is a sample configuration file. See the nxlog reference manual about the
## configuration options. It should be installed locally and is also available
## Please set the ROOT to the folder your nxlog was installed into,
## otherwise it will not start.
define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log

<Extension syslog>
        Module        xm_syslog
</Extension>

<Extension json>
   Module      xm_json
</Extension>

<Input in>
   Module      im_msvistalog
   Query       <QueryList>\
                <Query Id="0">\
                    <Select Path="Security">*</Select>\
                </Query>\
              </QueryList>
</Input>
<Output out>
   Module      om_tcp
```

```
   Host       192.168.56.101
   Port       5000
   Exec       to_json();
</Output>
<Route 1>
   Path       in => out
</Route>
```

Appendix B: Logstash Configuration

```
input {
  tcp {
    codec => json_lines { charset => "UTF-8" }
    # codec => json_lines { charset => CP1252 }
    port => 5000
    tags => [ "tcpjson" ]
    type => "nxlog"
  }
}
filter {
  if [type] == "nxlog" {
    json {
      source => "message"
    }
    if [SourceModuleName] == "eventlog" {
      mutate {
        replace => [ "message", "%{Message}" ]
      }
      mutate {
        remove_field => [ "Message" ]
      }
    }
    date {
      locale => "en"
      # timezone => "Etc/GMT"
      # timezone => "Asia/Shanghai"
      match => [ "EventTime", "YYYY-MM-dd HH:mm:ss" ]
    }
```

```
  }
}

output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
  }
}
```

Appendix C: Installation of Elasticsearch, Logstash and Kibana

Pre-requesites

Ubuntu 16.04 64 bit server with 4GB of RAM

Install Java.

Elasticsearch and Logstash require java to run.

Add java repository to the apt sources.list file.

```
sudo add-apt-repository -y ppa:webupd8team/java
```

Update the apt service to capture new source for java.

```
sudo apt-get update
```

Install Java from repositories.

```
sudo apt-get -y install oracle-java8-installer
```

Install Elasticsearch.

```
sudo apt install -y elasticsearch
```

Edit Elasticsearch configuration.

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

Uncomment the network.host line, then save and exit the file.

```
network.host: localhost
```

Restart Elasticsearch service.

```
sudo systemctl restart elasticsearch
```

Set the Elastic search service to run at boot.

```
sudo systemctl daemon-reload

sudo systemctl enable elasticsearch
```

Install Kibana

```
sudo apt-get -y install kibana
```

Edit the Kibana configuration file

```
sudo nano /opt/kibana/config/kibana.yml
```

Edit the server.hot value from 0.0.0.0 to localhost

```
server.host: "localhost"
```

Enable Kibana to start at boot after reboot.

```
sudo systemctl daemon-reload

sudo systemctl enable kibana

sudo systemctl start kibana
```

Install Logstash.

```
sudo apt-get -y install logstash
```

Enable Logstash to start after a reboot.

```
sudo systemctl restart logstash

sudo systemctl enable logstash
```

Create a file for Logstash to receive and process windows events from nxlogs by putting the configuration found in Appendix B.

```
sudo nano /etc/logstash/conf.d/windows.conf
```