

---

**Electronic Theses and Dissertations**

2018

# A Platform for analyzing log files using temporal logic approach: a test case with web server logs

Peris N. Muema  
*Faculty of Information Technology (FIT)*  
*Strathmore University*

Follow this and additional works at <https://su-plus.strathmore.edu/handle/11071/5990>

## Recommended Citation

Muema, P. N. (2018). *A Platform for analyzing log files using temporal logic approach: a test case with web server logs* (Thesis). Strathmore University. Retrieved from <https://su-plus.strathmore.edu/handle/11071/5990>

This Thesis - Open Access is brought to you for free and open access by DSpace @Strathmore University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DSpace @Strathmore University. For more information, please contact [librarian@strathmore.edu](mailto:librarian@strathmore.edu)

# **A Platform for Analysing Log Files Using Temporal Logic**

## **Approach: A Test Case with Web Server Logs**

**Muema, Peris Ndululu**

**066275**

**A Dissertation Submitted in partial fulfilment of the requirements for the Degree of  
Master of Science in Information Systems Security at Strathmore University**

**Faculty of Information Technology**

**Strathmore University**

**Nairobi, Kenya**



**April, 2018**

## DECLARATION

I declare that this work has never been previously submitted and approved for the award of a degree by Strathmore University or any other university. To the best of my knowledge and belief, this dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

© No part of this dissertation may be reproduced without permission of the author and Strathmore University

Student Name     **Muema, Peris Ndululu**

Student Number     **066275**

Signature     .....

Date     .....

### Approval

This dissertation of Muema Peris Ndululu was reviewed and approved by the following:

Supervisor Name     **Dr. Petr Matousek**

Lecturer, Faculty of Information Technology

Brno University

Signature     .....

Date     .....

## ACKNOWLEDGEMENT

I thank God Almighty for giving me the opportunity, strength and guidance to study Masters of Science in Information System Security at Strathmore University. My sincere thanks to my supervisor, Dr. Petr Matousek for his continual willingness to guide, understand and provide constructive feedback; to Dr. Joseph Sevilla, for his guidance and support as well throughout my course work.

I also acknowledge my family for their encouragement and prayers and my classmates Rachael, Collins and David for the team work and encouragement we have had during our coursework period.





## DEDICATION

I dedicate this dissertation work to God Almighty for strength, fit mind and good health throughout my studies. To my loving parents Mr. Benson K. Muema and Mrs. Anne W. Muema for their continual prayers, support and good will throughout my studies. To those who stood by me; my brothers David Kilonzo, Emmanuel Mbiuki and Simeon Ngugi, May God richly bless you. To my supervisors, Dr. Joseph Sevilla and Dr. Petr Matousek, who greatly assisted and guided me. I give special thanks to my dear friends and colleagues for their motivation and encouragement.



## ABSTRACT

Web logs are a set of recorded events between clients and web servers. Information provided by these events is valuable to computer system administrators, digital forensic investigators and system security personnel during digital investigations. It is important for these entities to understand when certain system events were initiated and by whom. To achieve this, it is fundamental to gather related evidence to the crime from log files. These forensic procedures however pose a major challenge due to large sizes of the web log files, difficulty in understanding and correlating to attack patterns associated to digital crimes. The connections of events that are remotely positioned in the large log files require extensive computational manpower.

This dissertation proposes the design, implementation and evaluation of a web log analysis system based on temporal logic and reconstruction. The case study will be on web server misuse. Temporal Logic operators represent system changes over time. The reconstruction of records in web server log files as streams will enable the implementation of temporal logic on the streaming data. The web server attack patterns established will be described by a special subset of temporal logic known as MSFOMTL (Many Sorted First Order Metric Temporal Logic). The attack patterns will be written in a special EPL (Event Processing Language) as queries and be parsed through Esper, a Complex Event Processing (CEP) engine. To ensure the proposed system increases the quality of log analysis process, log analysis will be performed based on a time window mechanism on sorted log files.

***Keywords: web server log, log analysis, web server misuse, misuse patterns, complex event processing, Esper***

## TABLE OF CONTENTS

DECLARATION .....	i
ACKNOWLEDGEMENT .....	ii
DEDICATION.....	iii
ABSTRACT .....	ii
LIST OF FIGURES.....	<b>Error! Bookmark not defined.</b>
LIST OF TABLES .....	viii
LIST OF ABBREVIATIONS .....	ix
DEFINITION OF TERMS .....	x
CHAPTER 1: INTRODUCTION.....	1
1.1 Introduction .....	1
1.2 Background of the Study .....	1
1.2 Problem Statement .....	3
1.3 Research Objectives .....	3
1.4 Research Questions .....	4
1.5 Scope and Limitation of the Study .....	4
1.6 Research Relevance.....	4
CHAPTER TWO: LITERATURE REVIEW.....	5
2.1 Introduction .....	5
2.2 Apache Web Server Log Files .....	5
2.2.1 Error Logs .....	5
2.2.3 Access Logs.....	6
2.3 Attacks on Apache .....	6
2.3.1 SQL Injection Attacks.....	7
2.3.2 Brute Force Attack.....	7

2.3.3 Command Injection Attack.....	8
2.3.4 Apache Vulnerabilities .....	8
2.4 Log Analysis.....	8
2.4.1 AWStats .....	9
2.4.2 Web Log Expert.....	9
2.4.3 Sawmill .....	10
2.4.4 PyFlag.....	10
2.4.5 Webalizer .....	11
2.5 Temporal Logic Overview .....	12
2.6 Related Work.....	13
2.6.1 Focus on Analysis of Large Log Files.....	13
2.6.2 Addressing a Variety of Log Formats .....	14
2.6.3 Correlation of Events through Log Entries .....	15
2.7 Conclusion.....	16
CHAPTER THREE: METHODOLOGY.....	17
3.1 Introduction .....	17
3.2 System Development Methodology .....	17
3.3 System Analysis.....	19
3.3.1 Feasibility Study .....	19
3.3.2 Research Design .....	19
3.4 System Design .....	19
3.5 System Implementation .....	20
3.6 System Testing.....	21
3.7 System Evaluation.....	22
3.8 Conclusion.....	22

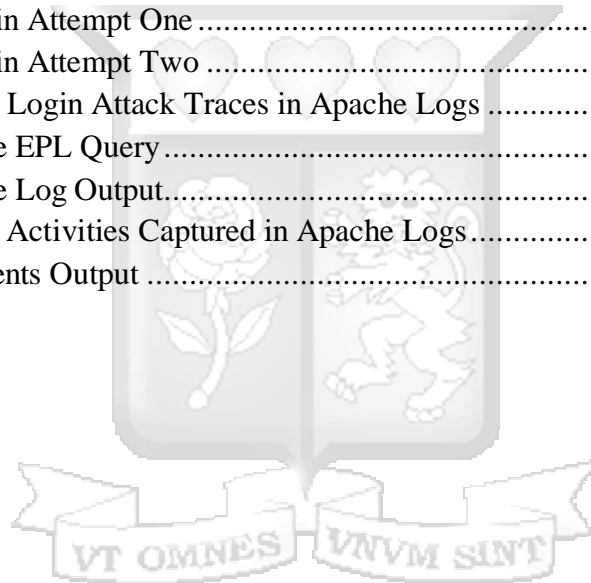
CHAPTER FOUR: SYSTEM DESIGN AND ARCHITECTURE.....	23
4.1 Introduction .....	23
4.2 System Architecture .....	23
4.2.1 Web Server.....	24
4.2.2 Web server Logs.....	24
4.2.3 Web Attack Patterns.....	26
4.3 Requirement Analysis .....	28
4.3.1 Functional Requirements.....	28
4.3.2 Non-functional Requirements.....	29
4.4 System Design .....	29
4.4.1 Context Diagram.....	30
4. 5 System Model .....	30
4.5.1 Use Case Diagram .....	30
4.5.2 Sequence Diagrams.....	34
CHAPTER FIVE: SYSTEM IMPLEMENTATION AND TESTING .....	36
5.1 Introduction .....	36
5.2 System Specification .....	36
5.3 System Implementation and Testing.....	36
5.3.1 Web Server Configuration.....	36
5.3.2 Web Log Analysis Configuration .....	37
5.4 System Features .....	40
5.4.1 Viewing Apache Access Logs .....	40
5.4.2 Creating Web Server Logs Datasets.....	40
5.4.3 Loading Datasets to Web Log Analysis System .....	42
5.4.4. Creating EPL Requests.....	42

5.4.5 Viewing EPL Responses .....	43
5.4.6 Identifying Web Attacks .....	44
5.5 System Testing.....	45
5.5.1 User Acceptance Testing .....	45
5.5.2 Unit and Integration Testing .....	49
CHAPTER SIX: DISCUSSION OF RESULTS.....	51
6.1 Introduction .....	51
6.2 Findings and Achievements .....	51
6.3 Review of Research Objectives.....	52
CHAPTER SEVEN: CONCLUSIONS AND RECOMMENDATIONS .....	53
7.1 Introduction .....	53
7.2 Conclusions .....	53
7.3 Recommendations .....	53
7.4 Future Work.....	54
REFERENCES .....	55
APPENDICES .....	59
Appendix A: Apache2 Web Server Installation .....	59
Appendix B: Esper Configuration in Eclipse IDE.....	60
Appendix C: Log.java class.....	61
Appendix D: Apache Log Input Adapter Configuration.....	64

## LIST OF FIGURES

Figure 1. 1: Web Server Usage 2016 - 2017 (Web Technology Survey, 2016).....	1
Figure 1. 2: Cyber Attacks Types 2016 (Calypix, 2016).....	2
Figure 1. 3: Web Application Vulnerabilities (Gordey, 2010) .....	2
Figure 2. 1: Web Application Attack Vector over HTTP 2016 (Stockley, 2016) .....	7
Figure 2. 2: PyFlag Overview Architecture (Cohen, 2008).....	11
Figure 2. 3: MSFOMTL Syntax (Gunestas & Bilgin, 2016) .....	12
Figure 3. 1: Agile Software Development Methodology (Harvin, 2016) .....	17
Figure 3. 2: Usage of Content Management Systems (World Wide Web Technology Survey, 2016) .....	20
Figure 3. 3: V-Model Testing Methodology (Borba & Cavalcanti, 2007) .....	22
Figure 4. 1: Web Log Analysis System Architecture.....	23
Figure 4. 2: SQL Injection Attack using GET method Pattern .....	27
Figure 4. 3: Brute Force Attack Pattern .....	28
Figure 4. 4: Web Log Analysis Context Diagram .....	30
Figure 4. 5: Use Case Diagram for Web Server Log Analysis Platform .....	31
Figure 4. 6: Sequence Diagram for Data Collection .....	35
Figure 4. 7: Sequence Diagram for Log Analysis.....	35
Figure 5. 1: Apache2 Start Service .....	37
Figure 5. 2: Web Server Logs Directory Location.....	37
Figure 5. 3: Web Server Logs CSV Format .....	37
Figure 5. 4: Log Parser Script .....	38
Figure 5. 5: Input Adapter Event Generation .....	38
Figure 5. 6: Input Adapter for- loop .....	39
Figure 5. 7: Input Adapter CSV Configuration .....	39
Figure 5. 8 : Web Log Analysis System Network Topology.....	39
Figure 5. 9: Apache Web Server Access Logs .....	40

Figure 5. 10: Apache Logs in Raw Format .....	41
Figure 5. 11: Log Data Conversion to CSV Command.....	41
Figure 5. 12: Apache Log CSV File .....	42
Figure 5. 13: Log CSV Loaded to Analysis System .....	42
Figure 5. 14: SQL Injection Attack Using GET method EPL Query .....	43
Figure 5. 15: SQL Injection Attack Using POST method EPL Query .....	43
Figure 5. 16: EPL Response.....	44
Figure 5. 17: SQL Injection using GET Method Traces in Apache Log .....	44
Figure 5. 18: SQL Injection Attack using GET method Log Output .....	45
Figure 5. 19 : WordPress Successful Login .....	46
Figure 5. 20: WordPress Login Traces in Apache Log .....	46
Figure 5. 21: WordPress Failed Login.....	47
Figure 5. 22: WordPress Failed Login Traces in Apache Logs .....	47
Figure 5. 23 : Failed Login Attempt One .....	48
Figure 5. 24 : Failed Login Attempt Two .....	48
Figure 5. 25: Brute Force Login Attack Traces in Apache Logs .....	48
Figure 5. 26 : Brute Force EPL Query.....	48
Figure 5. 27 : Brute Force Log Output.....	49
Figure 5. 28: Web Server Activities Captured in Apache Logs.....	50
Figure 5. 29: All Log Events Output .....	50





## LIST OF TABLES

Table 4. 1: Log Format Directive Percentages .....	25
Table 4. 2: View Apache Logs Use Case.....	32
Table 4. 3: Create Apache Logs Dataset Use Case .....	32
Table 4. 4: Load Datasets Use Case .....	33
Table 4. 5: Create EPL Requests Use Case.....	33
Table 4. 6: View EPL Responses Use Case .....	34
Table 4. 7: Identify Web Attack Use Case.....	34



## LIST OF ABBREVIATIONS

<b>AWS</b>	- Apache Web Service
<b>CEP</b>	- Complex Event Processing
<b>CLF</b>	- Common Log Format
<b>ELF</b>	- Extended Log Format
<b>EPL</b>	- Event Processing Language
<b>HTTP</b>	- Hyper Text Transfer Protocol
<b>IIS</b>	- Internet Information Services
<b>IPLoM</b>	-Iterative Partitioning Log Mining
<b>JSON</b>	- JavaScript Object Notation
<b>TL</b>	- Temporal Logic
<b>MSFOMTL</b>	- Many Sorted First Order Metric Temporal Logic
<b>NCSA</b>	- National Centre for Supercomputing Applications
<b>SIEM</b>	- Security Information and Event Management
<b>URL</b>	- Uniform Resource Identifier
<b>W3C</b>	- World Wide Web Consortium

## DEFINITION OF TERMS

**Log Files** – These are records of computer systems and applications events. Records are emitted by network devices, operating systems, programmable and application devices. This information is valuable to system and network administrators, digital investigators and other information security professionals.

**Log Analysis** -The process of examining log records. This may be done as procedures of system troubleshooting, security incident response, compliance to security and audit policies as well as understanding system and user interaction.

**Web Server** –A computer system that processes requests and provides responses via HTTP, a network protocol for distributing information over the World Wide Web.

**Web Sever Misuse** – This covers a range of activities such as hacking and intrusion which lead to server damage.

**Temporal Logic** – A system of rules for representing and reasoning about propositions qualified in terms of time. It is an extension of classical logic which includes operators dealing with time allowing formal specification of temporal events

**Misuse Patterns** – Comprise of signatures representing evidence indicating misuse. They correspond to activities being investigated.

**Complex Event Processing (CEP)** - Comprises of techniques for analysing streams of data regarding events in order to derive a conclusion. CEP analyses information to infer events or patterns to identify threats and respond to them.

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

This chapter gives a background study on web servers, web server logs, attacks targeting web servers as well as challenges experienced during web log analysis. The problem statement, specific research objectives of the study, related research questions, scope and limitations of the study including the research relevance of this study are discussed in this chapter.

## 1.2 Background of the Study

Apache is the most widely used Web server for commercial Web sites (Aulds, 2000). Figure 1.1 below indicates the percentages of websites using various web servers (Web Technology Survey, 2016).

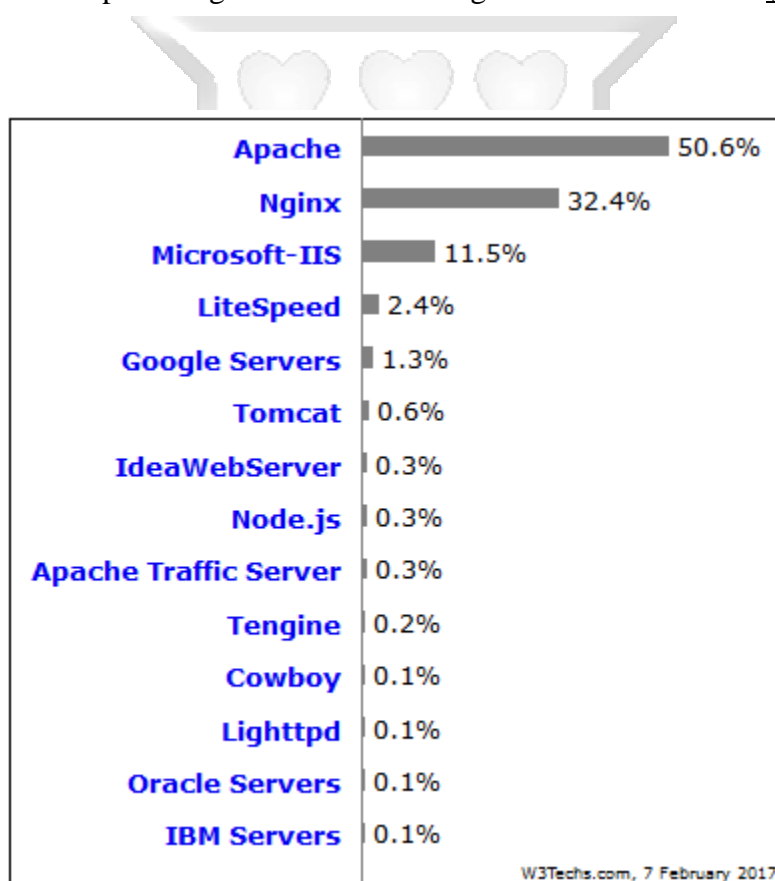


Figure 1. 1: Web Server Usage 2016 - 2017 (Web Technology Survey, 2016)

With the continual advances in technology, web servers have become an integral part of the fast developing enterprise environment today as they run and manage critical applications and provide valuable content to clients. Despite the tremendous progress in computer information security, complete secure computer systems and servers are still a challenge. In the year 2016, web application attacks composed of 24% of cyber-attacks (Calyptix, 2016).

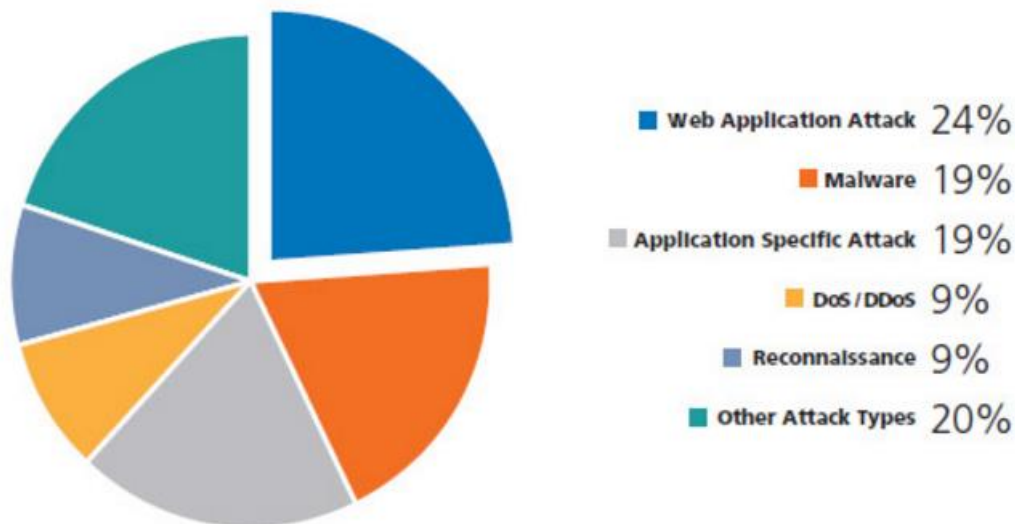


Figure 1. 2: Cyber Attacks Types 2016 (Calyptix, 2016)

Web servers are still prone to attacks through Cross Site scripts, information leakages and injection attacks. This is due to weak codes in programming and lack of web application structure sanitization (Kumar, 2016). Interest in web application security has risen dramatically relative to the number of vulnerable applications. The Open Web Application Security Project (OWASP) is an entity dedicated to improving the security of web application related software (OWASP Application Security Project, 2017).

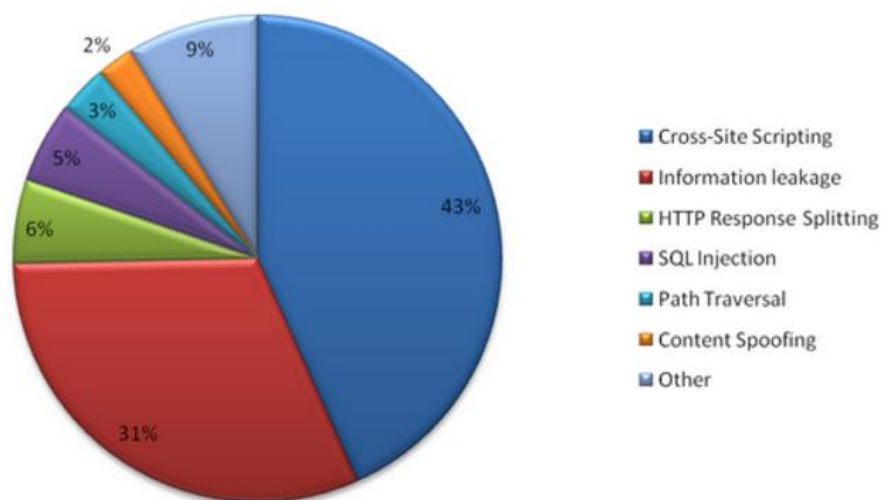


Figure 1. 3: Web Application Vulnerabilities (Gordey, 2010)

Log files are text like files that record users' activities on web servers. They reside inside the web server (Sharma & Gupta, 2013). Information about web server users such as user name, IP Address, timestamps, access request, referred URL, number of bytes and result status are contained in log files (Grace, 2011). These logs are useful in performing postmortem inspection of web attack incidents.

Web logs serve as essential sources of information describing web server traffic. The analysis of web log files containing crucial information is however a tedious task for forensic investigators due to challenges such as large size of the web log files, difficulty in understanding misuse patterns as well as complexities in correlating these patterns to the actual crime committed as this requires much effort in computation (Calzarossa & Massari, 2011). Current web log analysis tools such as AWStats, PyFlag and Sawmill are integrated with forensic analysis toolkits (Fry, 2011). In digital forensic investigations, real time analysis and monitoring of web logs is not possible. Log files are first extracted then analysed after (Singer & Bird, 2004).

## **1.2 Problem Statement**

With the increasing rate of crimes associated with digital systems, implementing digital procedures to obtain evidence has become vital. Web servers' events are stored in log files. These log files serve as important sources of evidence during web server misuse or crime incidents (Gordey, 2010). Existing systems used for the extraction and analysis of these evidence face major challenges due to the enormous sizes of web log files and complexities in understanding the attack patterns connected to the crime. This leads to slow log analysis which is time consuming. Currently, system network and security professional lack efficient standard platforms to define and share attack patterns in regards to log analysis.

## **1.3 Research Objectives**

This study is based on the following research objectives.

1. To study, understand and examine typical attacks on web servers focusing on HTTP.
2. To identify and generate log files of these attacks for analysis.
3. To review the existing systems available for web server log analysis.
4. To develop a web server log analysis platform used to define attack patterns using temporal logic.

5. To test the proposed solutions' capability in increasing the quality of log analysis to detect web attacks.

## **1.4 Research Questions**

This study is based on the following research questions.

1. Which attacks target web servers focusing on HTTP?
2. How are web server log files of these attacks identified and generated?
3. What are the existing systems available for web server log analysis?
4. How will the proposed web server log analysis platform be developed?
5. How will the platform be tested to proof its capability in increasing the quality of log analysis to detect web attacks?

## **1.5 Scope and Limitation of the Study**

This dissertation will only focus on Apache web server log data as the case study. Due to their publicity, these servers are mostly prone to attacks by hackers and malicious Internet users. In addition, WordPress Content Management System hosted on an Apache HTTP Server has been selected to aid in define misuse patterns. Log analysis will be limited to analysis of logs in the access.log files to detect SQL injection and brute force attacks specifically.

## **1.6 Research Relevance**

The collection of evidence in log files is not an easy task due to the large size of log files. Temporal logic capability of detecting complex patterns over streams in real time will increase the quality of the log analysis process. This system will highly benefit digital forensic investigators as it provides a library to share and store attack patterns previously detected and identified using a standard language formats while providing a fast accurate mechanism for large log file analysis.

## CHAPTER TWO: LITERATURE REVIEW

### 2.1 Introduction

This chapter focuses on the current web log analysis mechanisms studied and discussed by various researchers, their limitations and major challenges experienced. A review of different web server log files and evidence they contain as well as current web attacks and their impact is discussed.

### 2.2 Apache Web Server Log Files

There exists two types of Apache log files namely error and access logs. Access logs contain information related to client requests to web server and are used for analysing to traffic to the web server (Fry, 2011). Viewing the normal operation of the AWS within the access logs and error logs should determine if a problem exists through anomaly detection.

#### 2.2.1 Error Logs

Error log files are where the Apache HTTPD sends diagnostic data and records any errors that are encountered during request processing. Error logs are written to *error\_log* in UNIX systems and *error.log* in Windows. Diagnostic information logged in these logs include process startup and shutdown messages, critical event data, errors in request serving by the server having status codes between 400 and 503, and standards informational messages (Wainwright, 2008).

The format of events generated for error logs appear as:

**[ Day of Week / Month / Day / HH:MM:SS / Year ] [ LogLevel ] [ Hostname : Location (Originating Ip Address) ] [ Error Message ]**

Information contained in error logs has the correlation advantage. This greatly assists digital forensic investigators due to the fact that error events corresponding to request contain more information than those contained in access logs. In forensically determining an attack based on a specific module or the AWS, correlation of the version numbers within the error logs along with the attack string would be useful to an investigator. The more verbose the error logs directive is set to, the more likely development debug suggestions may offer additional information to an investigator (Fry, 2011).



### 2.2.3 Access Logs

These logs record all client requests sent to the web server. The Custom Log directive controls the location and contents of access logs. The log format directive is used to simplify the selection of log contents. The log format is specified and the logging can optionally be made conditional on request characteristics using environment variables. These access logs are formatted to three standards:

#### 1. Common Log Format

The World Wide Web Consortium, W3C, defines the Common Log Format, CLF, as the default log format used by UNIX web servers. A new log file is set up by the Custom Log directive using the defined common name. The configuration will write log entries in a format known as Common Log. Although this format is advantageous to system administrators, it poses a challenge to digital forensic system developers as tools used for development have to conform to a variety of user generated log files. A system administrator can add additional details as events are being recorded. An example to modify CLF is the Extended Log Format, ELF, also referred to as combined log file format (Aulds, 2000).

#### 2. Combined Log Format

The Combined Log Format is another frequently used format string. This format resembles the Common Log Format with an additional two fields (Fry, 2011).

#### 3. Multiple Access Logs

These logs are created by specifying multiple Custom Log directives in configuration file (Grace et al., 2011).

## 2.3 Attacks on Apache

Web servers are essential components of web-based applications. The Apache Web Server is often located at the border of a computer network hence making it vulnerable to attacks (Kumar, 2016).

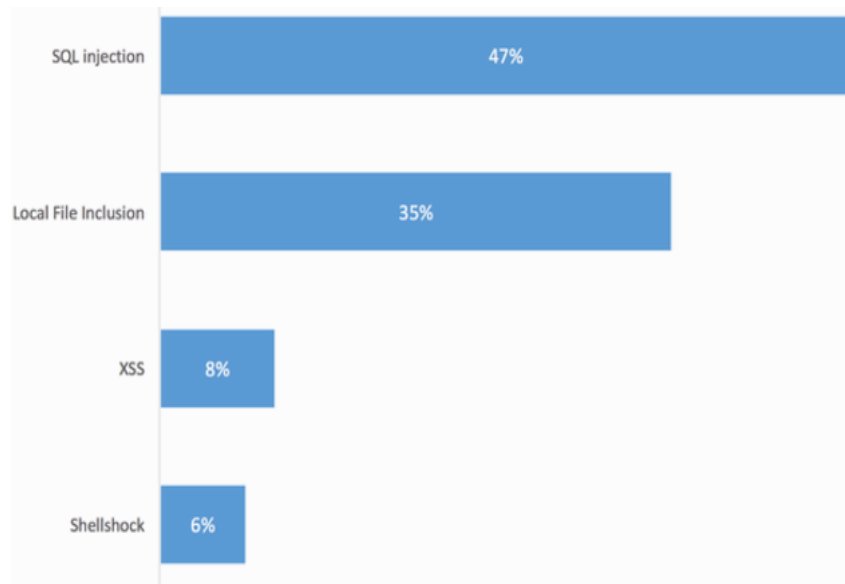


Figure 2. 1: Web Application Attack Vector over HTTP 2016 (Stockley, 2016)

### 2.3.1 SQL Injection Attacks

These attacks are carried out by inputting malicious inputs especially to data driven applications as interactive web sites. It consists of insertion of a SQL query via input data from a client to a web application. A successful attack may access sensitive data in databases, modify data as well as perform administration operations on the databases hence threatening the confidentiality, integrity and availability of these systems. An example, instead of submitting correct credentials in a web site authentication form, an attacker inputs the ['OR 1=1 --] and [ ] as username and password respectively (Halfold & Orso, 2006).

```
SELECT * FROM user WHERE login=" OR 1=1 --'AND pwd=' '
```

Using the single quote the attacker closes the SQL Where clause on login field enabling SQL injection of SQL control code into the query. The attacker then enters OR 1=1 which will evaluate to true -. The -- (double dash) operator marks the start of comments, prompting the SQL parser to ignore the Where clause on the password field (Janot & Zavarsky, 2008).

### 2.3.2 Brute Force Attack

This attack is an attempt to uncover passwords by continuously and systematically trying every combination of letters, symbols and numbers. The attacker sends these combinations to the server and analyses the responses until the correct combination that works is discovered (OWASP Brute Force Attack, 2016). Attackers use tools employed with wordlists and smart

rule sets to intelligently guess authentication passwords (Sowmya & Kumar, 2013). This attack majorly targets web sites which require user authentication. This causes a risk to user accounts as well as unnecessary traffic to the web server. Brute-force attacks also try to discover hidden web pages with the web application. Various HTTP brute force tools relay requests via a list of open proxy servers, since each request comes from different IP addresses, it may be difficult to prevent this attack by blocking the IP address.

These attacks are sent via the GET and POST methods as requests to the web server. An attacker may take a wordlist of known web pages and request each known web page then analyse the HTTP response code in order to determine if the web page exists on the targeted web server.

### **2.3.3 Command Injection Attack**

The goal of this attack is to inject and execute arbitrary system commands specified by attacker in a vulnerable web application. These injection attacks are possible as the vulnerable web application executes improper user supplied data to a system shell. User supplied data may include forms, cookies and HTTP headers. Arbitrary system commands are executed with the same privileges and environment as the vulnerable web application. Such attacks are possible due to lack of sufficient input data validation (OWASP Command Injection, 2016).

### **2.3.4 Apache Vulnerabilities**

Programming errors have security implications, the errors can be exploited to misuse system resources hence pose as vulnerability. In 1998, a programming error in Apache, small sized requests caused Apache to allocate large memory. Vulnerabilities such as non-exploitable buffer overflows may cause the Apache server to crash when attacked. Since Apache runs in pre-fork mode, many instances of the server run in parallel too. If the child crashes, the parent process creates a new child. An attacker will send multiple requests to disrupt the server's operations. In a multithreaded mode of operation, there exist one server process hence crashing will result to shut down of the whole server making it unavailable (Ristic, 2005).

## **2.4 Log Analysis**

The advancement of digital evidence extraction from digital devices is continually increasing in complexity due to new systems developed over time. With the large storage capacity in devices, the partitioning of memory for evidence analysis enable current forensic tools provides

relevant information to forensic investigators. This greatly minimises the amount of data analysed. Most software applications, operating systems and system services are able to store log data about events occurring during normal operations due to their development process. The aggregation of log events is crucial to a forensic investigator during reconstruction of the crime incident. The analysis of data in the log files is accurately reconstructed to display user activities. Although log files are not specifically for the purpose of forensic investigations, they contain important evidentiary information and provide hints to other sources that may contain vital information on an incident as well (Fry, 2011). This section provides an overview of some of the current tools which exist and aid in performing enhancing the log analysis process.

#### **2.4.1 AWStats**

Advanced Web Statistics, AWStats is a free software distributed under the GNU General Public License v3 (Destailleur, 2015). It is a Perl-based open source log analyser which creates advanced web, mail, ftp and streaming server statistical reports based on information contained in server logs. The resulting data is presented in clearly visible web pages. This tool can be run via a web browser's common gateway interface or from an operating system command line directly. AWStats is capable of processing large log files from Apache, Microsoft IIS and WebStar through the use of intermediary data base files. Examples of analysed data presented in the reports include most visited sites, authenticated user visits, domains, countries of visitor's hosts, latest visits and unresolved IP address lists, HTTP errors and referrer search engine. AWStats analyses the AWS log files in CLF, ELF, MS IIS W3C, other FTP, mail server, proxy and streaming media. A user may also define their log formats for analysis; however data gathered is limited to data in the log files themselves. This tool is neither designed as a log file correlation engine nor a forensic tool but for the purpose of analysis of web related log files. This data presented in the reports greatly assist in investigations. Addition of correlation capabilities and support for other log files will provide necessary characteristics for forensic web analysis. The AWStats software is available on the website link: <http://www.awstats.org/> (Last accessed 7th February 2017).

#### **2.4.2 Web Log Expert**

It is a software application that analyses individual web server log files such as Apache and IIS, and generates reports corresponding to specific web pages (Tyagi & Choudhary, 2015). It is Windows-based and provides detailed information on site's visitors. The results include general statistics, accessed files, and statistics about paths taken through the site, information

about referring pages, search engines, browsers, operating systems and errors. The built-in wizard aid in profile creation of a specific site in order to analyse it, this is mandatory in performing log analysis. After analysis, a HTML file will be present precise information on total hits, average page views per day, graphs displaying daily visitors, top hosts and daily referring sites. The Weblog Expert application is available on the website link <https://www.weblogexpert.com/> (Last accessed 7th February 2017).

### **2.4.3 Sawmill**

The main advantage of this tool is, it includes plugins that automatically detect over 800 different types of log files and provides methods for plugin definition for nonstandard log file types (Fry, 2011). Sawmill architecture includes the log importer, sawmill database, web server, reporting interfaces, command line interfaces, scheduler and data manipulation languages for log analysis. Salang, sawmill language is used in displaying pages defining log filters in terms of regular expressions and conditional logic. This tool provides the access necessary to perform correlation between log data sources. The Sawmill tool is available on the website link <https://www.sawmill.net/> (Last accessed 7th February 2017).

### **2.4.4 PyFlag**

It is an open source web based application that performs log analysis through an extensive Graphical User Interface. It is capable of analysing large volumes of log files from disks, images and network traffic such as tcpdump. The data is added to MySQL database for faster querying but log types are specified by user since this tool only views log files hence an analyst must have prior knowledge and experience to perform the required log analysis. Although regular expression can be considered as inputs, this tool does not contain prebuilt analysis capabilities (Cohen, 2008). The PyFlag application is available on the website link <http://pyflag.sourceforge.net/Downloads/> (Last accessed 7th February 2017).

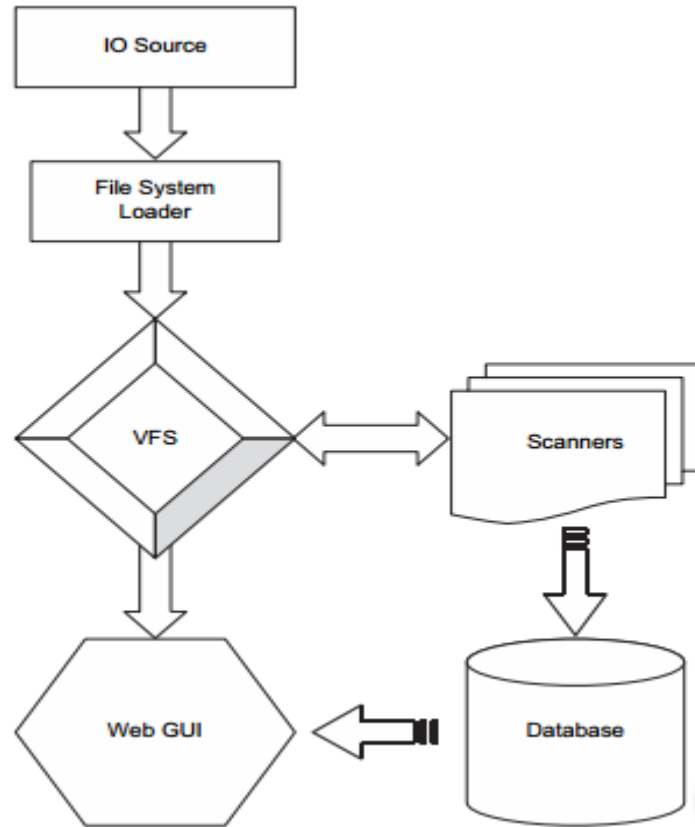


Figure 2. 2: PyFlag Overview Architecture (Cohen, 2008)

#### 2.4.5 Webalizer

This is a fast and free web server log file analysis program which produces detailed usage statistics in HTML format for viewing with a standard web browser (Barret, 2014). The code is written in C language hence portable in Linux, Solaris and UNIX operating systems. Log formats that are analysed include CLF, variations of NCSA combine log format, xferlog CLF logs and W3C ELF. In addition, this tool has the capability of decompressing bzip2 and gzip compressed logs without the need for uncompressing hence saving on memory space thus providing analysis compatibility for larger log files. The log analysis carried out by this tool provides summary statistical information on the web server by sites or time based reports presented in graphical or tabular formats that are configurable from command line. This tool does not run in real time, provides no correlation capabilities and does not automatically detect log file types. The Webalizer program is available on the website link <http://www.webalizer.org/> (Last accessed 7th February 2017).

## 2.5 Temporal Logic Overview

The development of formal methods for automatic verification and specification of real time computer systems has been steadily increasing Ahmed & Lisitsa (2011). System satisfaction based on its specifications is proofed through these verification techniques. Verification techniques aid in ensuring that a system execution satisfies a specific property. Logical -based formal methods applied in runtime verification techniques provide concise procedures to formally represent system specifications and provide the necessary mechanisms to reason about given properties

The main objective of logic in computer science is to create languages modeling situations encountered by computer system professionals in ways they can be formally reasoned. Temporal logic is an extension of classical logic which includes operators dealing with time allowing formal specification of temporal events. Quantitative temporal properties are vital in dealing with real-time systems. Koymans (1990) introduced the Metric Temporal Logic where the qualitative temporal operators are converted to metric temporal operator (quantitative). The conversion from qualitative to quantitative temporal operators is carried out by Metric Temporal Logic through constraining the temporal operator with bounded or unbounded interval. The metric extension to temporal logic is useful in relating events in real- time systems.

A special subset of temporal logic is Many Sorted First Order Metric Temporal Logic (MSFOMTL) which aids in modeling attack patterns for analysing log records that change overtime. MSFOMTL syntax is:

$$\begin{aligned} \varphi ::= & \top \mid \perp \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \\ \rightarrow & \varphi \mid (\forall x)\varphi \mid (\exists x)\varphi \mid \tau \mid \diamond_{[t_1, t_2]} \varphi \mid \square_{[t_1, t_2]} \varphi \mid \\ & \blacklozenge_{[t_1, t_2]} \varphi \mid \blacksquare_{[t_1, t_2]} \varphi \mid R_{[t_1, t_2]}^n \varphi \end{aligned}$$

Figure 2. 3: MSFOMTL Syntax (Gunestas & Bilgin, 2016)

Where  $p$  is any propositional atom from some atoms and the symbols  $\top \mid \perp \mid$  are MSFOMTL formulas. Symbols are grouped into logical and non-logical symbols. Logical symbols are

quantifiers  $\forall$  and  $\exists$ , the logical connectives  $\wedge$ ,  $\vee$  and  $\neg$ , the logical binary predicate symbols  $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$ , and  $\geq$ . The bounded future temporal operators  $\Diamond_{[t_1, t_2]}$  (“eventually”), and  $\Box_{[t_1, t_2]}$  (“always”), the past bounded temporal operators  $\Diamond_{[t_1, t_2]}$  (“sometimes in the past”), and  $\Box_{[t_1, t_2]}$  (“always in the past”) and  $R^n_{[t_1, t_2]}$  means  $\Phi$  “repeats n times”. The subscripts  $[t_1, t_2]$  in the operators refer to their scope (between the moments  $t_1$  and  $t_2$  from now) (Gunestas & Bilgin, 2016).

## 2.6 Related Work

Log analysis has been an active field of study over time, it involves the processes of inspecting computer system records in order to identify attacks and mitigate risks. Various researchers have developed and implemented different approaches for the purposes of log file analysis. Studies have shown that various log analysis approaches exist with different levels of success. This research can be grouped into three main categories:

1. Focus on analysis of large log files
2. Addressing a wide variety of log formats
3. Correlation of events through log entries.

### 2.6.1 Focus on Analysis of Large Log Files

Vernekar and Buchade (2013) proposed a system based on Map Reduce algorithm for log analysis which provides appropriate security alerts. Map Reduce is a distributed system algorithm which uses clusters of computer as resources. The Hadoop framework is the most popular implementation of Map Reduce algorithm. This algorithm consists of Map phases and Reduce phases. The log file to be processed is provided as an input to the Map phase where it is divided into 16Mb to 128Mb chunks. These chunks are then distributed to several Map functions residing on the nodes of the Hadoop clusters allowing parallel processing of various file chunks generating the intermediate key value pair output faster. Each Reduce function is associated with a key. The intermediate output produced by the each Map phase is assigned to the Reduce phase, where Reduce functions are given the values which belong to the particular key. The Reduce function will then provide the final result or log report.

The Iterative Partitioning Log Mining approach proposed by Makanju & Milios (2009) is another system towards log analysis of large log files. It is an algorithm for dividing event logs



into clusters and mining the patterns for alert generation based on the patterns. It works by partitioning a set of log events through four iterative steps.

- i. Partition by token count.
- ii. Partition by token position.
- iii. Partition by search for bijection.
- iv. Discover cluster descriptions or line formats

At each step the resulting partition get closer to containing log messages only produced by the same line format. The fourth steps attempt to identify the line formats that produced the lines in each partition, these discovered partitions and line formats are the output of the algorithm

Havens and Lunt (2012) focuses on the efficiency of three off the shelf Bayesian spam filters through classification of log entries. The Filter Effectiveness Scale is used to compare the filters. The filters are first tested with Spam Assassin corpus, then they are tested for their ability to differentiate two types of log entries taken from actual production systems and finally the filters are trained on log entries from actual system outages and then tested on effectiveness for finding similar outages via the log files.

Kalamatianos and Matthews (2012) proposed a log analysis technique aimed to aid digital analysts compute smaller collection of events related to their analysis objective rather than the entire original large log file. This technique is based on computing a similarity score between logged events and a group of significant events known as beacons. The beacon events are selected through an automated process which searches for unusual events either by operators auditing system logs or hypothesis generation process. Being a domain independent approach, domain knowledge is not utilized. Different types of events are not treated in a preferential manner while computing the similarity score, therefore collection of log data conforming to different formats and schemas is possible. A pre-existing training data set is not required as this limits most approaches based on machine learning techniques.

### **2.6.2 Addressing a Variety of Log Formats**

Software log analysis heavily contributes to software testing and troubleshooting. The first step in automating log analysis is extraction of data. Jayathilake (2011) introduced a new log

data extraction generic scheme that provides advanced features to hasten log file analysis procedures. It has the capability of handling various types of log files such as xml, tabular, line and binary logs with complex structures and syntax. Log entries are differentiated using attributes such as length, minimum length, maximum length, delimiters and possible values. The output consists of a tree structure resembling a mind map containing information related to a specific case.

The information in computer system logs is crucial for gathering forensic evidence when investigating system attacks. Arasteh and Debbabi (2007) proposed a model checking approach for addressing the issue of formal analysis of logs. The logs are modeled as a tree with labeled log events. Each event is expressed as an algebraic term providing structure to these events. Signatures of the algebraic term are selected to include relevant information required to conduct the analysis process.

### 2.6.3 Correlation of Events through Log Entries

In today's technology Security Information and Event Management, SIEM systems aid in gathering information from network devices, software and hardware security systems and applications. Event and activity monitoring and reporting capabilities of SIEM systems can be used in real time correlation of events with other conceptual information. The widely used SIEM products include Splunk which improves the detection and response to advanced threats through the Splunk User Behavior Analytics which provides broad security intelligence, LogRhythm which unifies SIEM, log management and network endpoint monitoring and forensics with advanced security analytics and ArcSight which provides big data security analytics and intelligence for security information and log management by collecting security log data from operating systems and applications and analysing them for patterns of attacks or malicious activity.

OSSIM, an open source SIEM system product provides complete event collection, normalization and rule- based correlation. It allows users define the dependencies between events via xml file instead of using temporal logic. On the other hand, MASSIF which is based on Complex Event Processing translates OSSIM's directives into complex event processing queries that run in parallel (Vianello, et al., 2013)

Complex Event Processing is a technique involves collecting events from different sources, filtering, and transforming, detecting patterns, correlating and aggregating them to complex

event. These systems employ temporal logic to some extent via various event processing languages in their processes. Esper and StreamBase are examples of Complex Event Processing frameworks used today together with event processing languages as EventFlow and StreamSQL (Albek & Bax. 2005).

Ahmed et al. (2011) implemented MSFOMTL to define misuse patterns which are transformed to StreamSQL queries and run in Streambase platform. (Herrerías & Gómez, 2010) designed an Automated Forensics Diagnosis System that reconstructs attacks after incidents to carry out log analysis with the event correlation module. Therefore, the system detects multi-step attacks reducing false positives. The main objective of this system is reducing time required to search for digital evidence by forensic investigators as well as eradicating complexities involved in this process.

## 2.7 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.

Similar to research done on improving the analysis of large log files, the proposed approach based on temporal logic and log reconstruction serves as a solution towards large log files analysis by minimizing the investigation scope using time windows. The Many Sorted First Order Metric Temporal Logic (MSFOMTL), a unique case of temporal logic will be used in this case study as it allows for the specification of packets arrival time and enables representation of attacks formally and conduct runtime verification to detect their occurrences in the stream of incoming events.. It will be applied in modeling attack patterns for analysing log events over time periods.

## CHAPTER THREE: METHODOLOGY

### 3.1 Introduction

A methodology involves a systematic approach to a resolution of an existing problem. It offers a theoretical understanding of which methods can be applied to specific cases to produce specific results (Irny & Rose, 2005). This chapter is concerned with describing the methodology that will be employed to enable the proposed system answer the research questions outlined in Chapter 1. In this chapter the System Design and Analysis methods, Implementation methods, Testing and Evaluation methods are discussed.

### 3.2 System Development Methodology

The system design method used in this case study was the agile software development methodology. It provides opportunities to assess the direction of a project throughout the development lifecycle (Harvin, 2016).

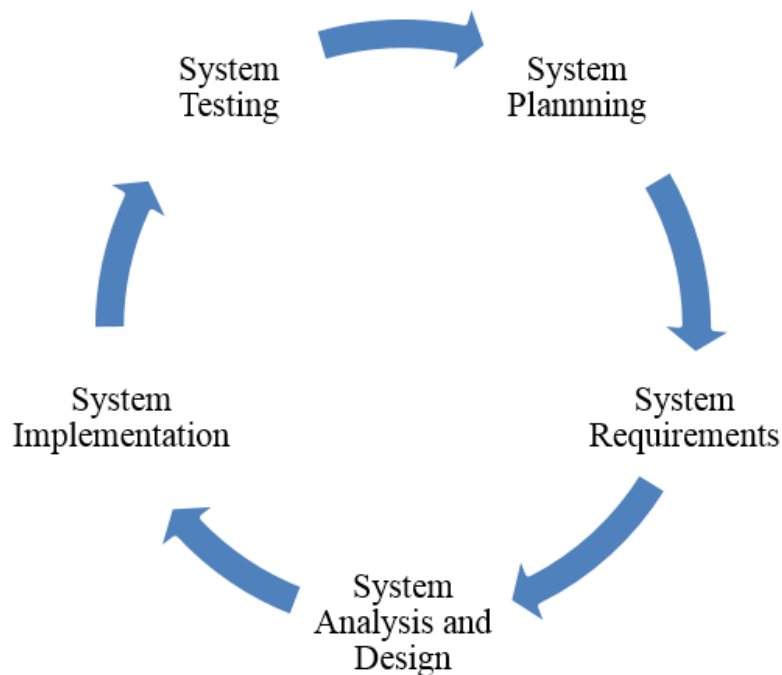


Figure 3. 1: Agile Software Development Methodology (Harvin, 2016)

In this methodology, each system task was allotted a time slot ensuring delivery of specific features for each system release based on the previous system functionality. The system was developed in incremental iterations or cycles (Szalvay, 2004). Each system release was tested ensuring continuous system quality. The following are the steps involved in system development process:

- **Planning:** This step involves outlining the necessary procedures required to achieve this study's research objectives. This included reviewing how web log analysis has been addressed by various researchers and existing systems and tools used to perform web log analysis as well as their merits and limitation.
- **System and Requirement Analysis:** This involves analysis of data collected so as to identify requirements needed for system development. This included emulating web attack and extracting the web logs in preparation for analysis. Once the system requirements have been understood, a system design can be done determining the system structure and system functionalities.
- **System Design:** The system requirements were translated into a technical description and design. It involved a web log analysis system architecture which was modelled in various design diagrams as use cases and sequence diagrams.
- **System Implementation:** This included the development of the actual system dependent on the design diagrams produced in the previous phase. This was achieved through extraction of web logs from Apache web server and analysis using Esper Complex Event Processing engine running on a Linux distribution.
- **System Testing:** The system developed was presented to various system administrators for use. Sample web logs were loaded to the Esper, CEP engine as events for analysis. This aided in determining if the system functionalities satisfy the study's research objectives, Feedback was collected and was used for further enhancements in the system under development.
- **System validity and reliability** was checked by evaluating the success rate of tests during system requirement analysis. This was done to ensure he developed system answered the outlined research questions of this study.

### 3.3 System Analysis

The system requirements were identified and analysed. The results were used to answer the research questions as well as aid in designing the web log analysis system.

#### 3.3.1 Feasibility Study

This was conducted through literature review on existing and other proposed methods used for web server log analysis.

#### 3.3.2 Research Design

The research design used was dependent on the specific research objectives and had to relate with the research questions (Kothari, 2004). It serves as a road map indicating how a researcher goes about answering specific case study research questions (Bryman & Bell, 2007).

The quasi-experimental research design will be used for this test case. It tests the relations in given environments with the aim of analysing outcomes of interest based on treatments (Levy & Ellis, 2011). This research design has been chosen as the research analyses if the proposed web log analysis using temporal logic approach enables system network and security administrators as well as other information security professionals identify and define web server misuse and attack patterns as well as accurately analyse large log file in a timely manner.

### 3.4 System Design

This 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). Use cases will be represented in texts and describe actions an actor can do in the system. In this system, the actors include a malicious web user who launches attacks on the web application and the forensic investigator who analyses the web server log files to extract evidence of the web attacks launched. Sequence and data flow diagrams will be used to illustrate how the developed system will handle different data flows between system processes. It aids in analysing the system to determine if the required system data and processes have been defined (Mohapatra & Joseph, 2014).

### 3.5 System Implementation

The system included the Apache web server, WordPress Content Management System hosted on the Apache server, Apache web server logs and Esper, a complex event processing (CEP) engine, to query the attack patterns.

#### i. Apache HTTP Server

Due to their publicity, these types of servers are mostly targeted by attackers and malicious web users. Web logs from these server will be extracted for analysis to identify the presence of web server attacks.

#### ii. Web Application

Widely available themes such as WordPress, Joomla, Drupal and Magento are used as web server development basis. Different web server activities built on a similar theme will leave same traces on log records. Thus, once patterns specific to a particular theme are defined, then it is possible to search for similar patterns indicating malicious events on other web sites built on the same theme (Gunestas & Bilgin, 2016). The diagram below indicates that 27.3% of all the websites use WordPress that is a content management system market share of 58.5%. 3.4% of all the websites use Joomla that is a content management system market share of 7.2% while 2.2% of all the websites use Drupal that is a content management system market share of 4.8%.

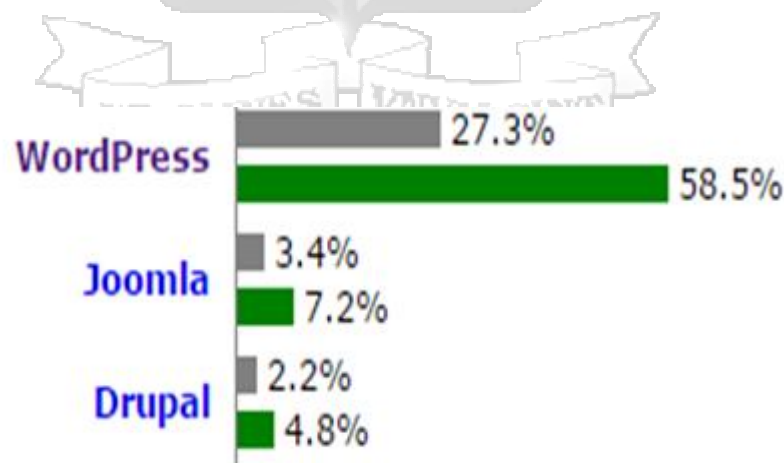


Figure 3. 2: Usage of Content Management Systems (World Wide Web Technology Survey, 2016)

In this dissertation, WordPress has been selected as the main focus of this case study to define misuse patterns. The web application developed will be based on WordPress theme. The hosting of the website will be done on an Apache HTTP Server.

### **iii. Complex Event Processing Engine**

Temporal logic, a form of modal logic is used widely in verifying the correctness of critical computer systems (Huth & Ryan, 2004). A special subset of temporal logic referred to as Many Sorted First Order Temporal Logic, MSFOTL, will be used to model web server attack patterns for analysing web logs which span over time. MSFOMTL entails special features that are efficient and concise in describing log events requiring investigation. After modelling the web attack patterns, the Event Processing Language, EPL, will be used to define these patterns into queries. A Complex Event Processing engine, Esper, will be used to query these patterns (Herreman, 2006).

## **3.6 System Testing**

The system was tested against its specifications to verify whether it complies with the functional requirements. The V-Model testing methodology also referred to the verification and validation model was employed for system testing. In the V-Model, development and quality assurance activities are done simultaneously (Borba & Cavalcanti, 2007). There is no discrete phase called Testing, rather testing starts right from the requirement phase. As illustrated in Figure 3.4 below, User Acceptance Testing, System Testing, Integration Testing and Unit Testing occur simultaneously as verification and validation activities go hand in hand.



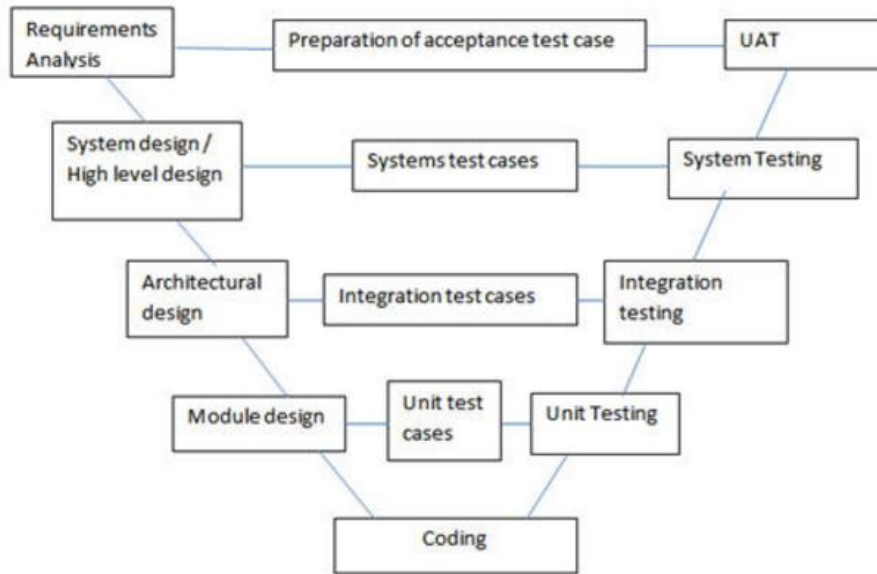


Figure 3. 3: V-Model Testing Methodology (Borba & Cavalcanti, 2007)

### 3.7 System Evaluation

The system will be evaluated by the developer to establish its validity and whether the study's research objectives are achieved. This is essential as it will indicate if the system developed will enhance web log analysis process.

### 3.8 Conclusion

This chapter has provided an overview of methods that will be used to ensure that the proposed system meets the research objectives as well as answering the research questions.

## CHAPTER FOUR: SYSTEM DESIGN AND ARCHITECTURE

### 4.1 Introduction

The web server log analysis system based on temporal logic and reconstruction was implemented with the sole aim of enabling network and system security administrators define web server attack patterns for fast and easier identification of the attacks. This chapters illustrates the system architecture, system design and the components used to implement the proposed web server log analysis system. Interaction diagrams have been used to aid in illustrating the interaction between users and the system as well as data flow between various system modules required for proper system function.

### 4.2 System Architecture

The web log analysis system comprises of a web server, web server logs, defined web attack patterns using temporal logic and a complex event processing engine used to query and analysis web attack patterns.

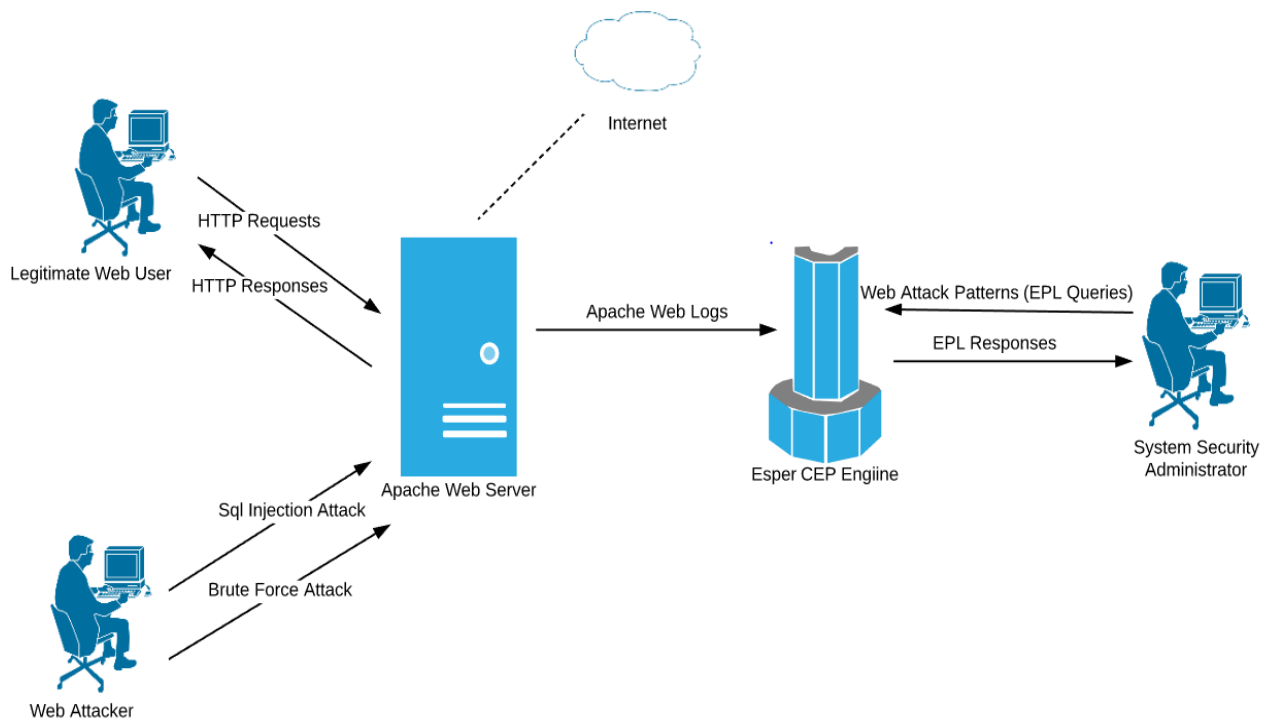


Figure 4. 1: Web Log Analysis System Architecture

### 4.2.1 Web Server

The web server selected for this system is Apache web server due to its ease of use as well as its vulnerability to most web attacks including SQL injection attacks, brute force attacks and remote shell injection attacks. Web logs from the Apache web server will be extracted for analysis and identification of these attacks.

### 4.2.2 Web server Logs

There exists two types of Apache log files namely access and error logs. Access logs contain information related to client requests to web server and are used for analysing to traffic to the web server (Fry, 2011). The system will specifically analyse the access logs.

#### I. Access Logs

These logs record all client requests sent to the web server. The Custom log directive controls the location and contents of access logs. The log format directive is used to simplify the selection of log contents. The log format is specified and the logging can optionally be made conditional on request characteristics using environment variables. These access logs are formatted to three standards:

#### A. Common Log Format

The log format is used by the AWS access logs, the format is indicated below (Aulds, 2000):

**[remotehost] [identd] [authuser] [date] [request URL] [status] [bytes]**

The **remotehost** indicates the IP address of the clients that sent request to the web server. The second field, **identd**, contains the identity of the visitor such as email address or any other unique identifier. The **authuser** field contains the clients' username credentials. This field appears only when the client requests a protected document requiring a user ID and password (Wainwright, 2008). The common format was modified to custom log file format as shown below:

**LogFormat "%h %l %u %t \" %r \" %>s %b" common**

**Example: 127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache\_pb.gif HTTP/1.0" 200 2326**

The format string consists of directive percentages, each informing the server to log a particular piece of information. The entries give details about a client who had made a request to the web server. Table 4.1 gives a detailed description of each directive in the log format (Grace, Maheswari, & Nagamalai, 2011). This system works mostly with this format.

Log Field	Directive	Example	Description
Remote Host	%h	(127.0.0.1)	IP address of client who made request to web server.
Identity	%l	(-)	The hyphen after the client's IP address indicates that the requested information is unavailable
AuthUser	%u	(frank)	This is the user ID of client requesting a document. It is determined by HTTP authentication.
Date	%t	( [10/Oct/2000:13:55:36 -0700])	This indicates the time and date. It resembles the format [day/month/year: hour: minute: second zone]
Request URL	\"%r\"	(\"GET /apache_pb.gif HTTP/1.0\")	This is the client's request in quotes. GET is the method used apache_pb.gif is the information requested by the client. The protocol used by the client is given as HTTP/1.0
Status	%>s	(200)	This is the status code sent by the server. The codes beginning with 2 for successful response, 3 for redirection, 4 for error caused by the client, 5 for error in the server
Bytes	%b	(2326)	The last entry indicates the size of the object returned to the client by the server, not including the response headers. If there is no content returned to the client, this value will be \"_\"

Table 4. 1: Log Format Directive Percentages

## B. Combined Log Format

The Combined Log Format is another frequently used format string as shown below.

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"  
" combined
```

This format resembles the Common Log Format with an additional two fields, each uses the directive percentage `%{header}i`, where *header* can be any HTTP request header.

[remotehost] [identd] [authuser] [date] [request URL] [status] [bytes] [referrer] [agent]

The referrer field indicates that the client request was generated by clicking a link from different page in another web site and header content shows the page URL. The agent field represents the browser used by the client, requester (Fry, 2011).

## C. Multiple Access Logs

They are created by specifying multiple Custom Log directives in configuration file. The configuration is as shown below.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

```
CustomLog logs/access_log common
```

```
CustomLog logs/referer_log "%{Referer}i -> %U"
```

```
CustomLog logs/agent_log "%{User-agent}i"
```

There are three files created as access logs containing client information. It is a combination of common log format and combined log format. The first line is basic common log format information and the second line is referrer and browser information (Grace et al., 2011).

### 4.2.3 Web Attack Patterns

Many Sorted First Order Temporal Logic, MSFOTML, a special subset of temporal logic is used to model the web server attack patterns. For illustration purposes,

$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$  where;

$X_1$ - represents client IP address.

$X_2$ - represent the host.

X<sub>3</sub>- represent the username or user ID.

X<sub>4</sub>- date-time variable representing the Timestamp.

X<sub>5</sub>- represents Request Method.

X<sub>6</sub>- represents Request URL

X<sub>7</sub> -represents Request code.

X<sub>8</sub>- represents sent bytes.

X<sub>9</sub>- represents the referrer.

X<sub>10</sub>- represents user agent.

There are four pattern types which are classified into;

i. Single Record Pattern

From a single log record, one can deduce a specific log activity hence defining a pattern representing corresponding activity. While defining these patterns, keywords or regular expressions matches are employed over log entry values. An example of such a pattern can be written to represent a SQL injection attack using GET method.

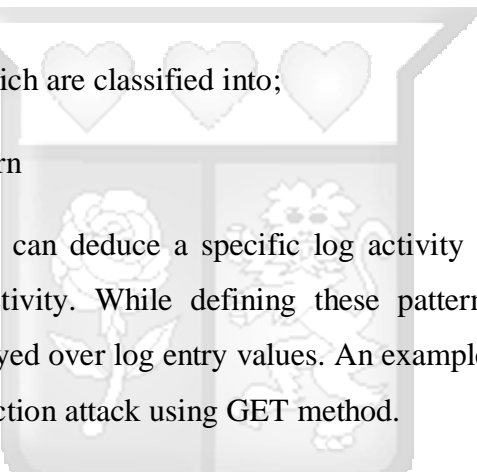

$$\begin{aligned} &(\exists x_1, x_2, x_3, x_4, x_8, x_9, x_{10}) \\ &P(x_1, x_2, x_3, x_4, "GET", x_6, 200, x_8, x_9, x_{10}) \\ &\wedge \text{contains}(x_6, "\% \text{ or } \% = \%") \end{aligned}$$

Figure 4. 2: SQL Injection Attack using GET method Pattern

ii. Multiple Record Patterns

These record patterns are used in representing complex events comprising of more than one events spanned over time that are related to each other. Not only are related multiple record patterns defined but also those corresponding to evidence of complex patterns by including multiple single record patterns. An example of such a pattern can be written to represent a brute force login attack against a WordPress login page which displays when unsuccessful login attempts occur.

$$(\exists x_1)R_{[0,30]}^{30} \left( \begin{array}{c} (\exists y_2, y_3, y_4, y_6, y_8, y_9, y_{10}) \\ P(x_1, y_2, y_3, y_4, "POST", y_6, 200, y_8, y_9, y_{10}) \\ \wedge \text{contains}(y_6, "wp - login") \end{array} \right)$$

Figure 4. 3: Brute Force Attack Pattern

### iii. Compound Record Patterns

It is possible to develop complex event patterns comprising of single and multiple record patterns in cases where single record patterns are inadequate in addressing log activities precisely. This aids in corroborating evidence of an activity that occurred in the system. Compound record patterns are defined in the event that single or multiple record patterns fail to differentiate an expected activity leading to false positives alerts.

### iv. Abstract Record Patterns

Some compound patterns have similarities hence can be abstracted from. After abstraction, general record patterns that address more system misuse activities are obtained.

## 4.3 Requirement Analysis

Based on the study's research objectives and the system user requirements, this section outlines various requirements that ought to be met by the system developed.

### 4.3.1 Functional Requirements

1. Connected to Apache2 web server and view logs related to HTTP requests and responses based on WordPress CMS.

- a. No duplication of web server access logs.
- b. Retrieved metadata of each access log.
- c. Log2CSV parser to convert log metadata to CSV form and store in CSV file with the following headers;
  - i. Host
  - ii. Log Name
  - iii. Date Time

- iv. Time Zone
- v. Method
- vi. URL
- vii. Response Code
- viii. Bytes Sent
- ix. Referrer
- x. User Agent

2. Created a CSV Input Adapter to send log data stored in CSV file as events to Esper CEP engine for analysis.

3. Performed analysis on each web log to identify web misuse cases based on defined web attack patterns.

#### **4.3.2 Non-functional Requirements**

##### **4.3.2.1 Software Requirements**

- i. Apache2 web server.
- ii. Simple web application based on WordPress theme. The web application is hosted on the Apache2 web server.
- iii. Log2CSV parser to convert the access log metadata to CSV form.
- iv. Eclipse IDE for Java Developers (Neon.3) for system code development.
- v. Esper 6.0.1, a Complex Event Processing engine used for log event analysis.
- vi. Java, a general purpose programming language for coding.
- vii. Windows 10 operating system

##### **4.3.2.2 Hardware Requirements**

- i. Machine: Intel Core i5 or higher
- ii. Clock speed and processor: 2.50GHz or higher
- iii. System Memory: 4 GB.

#### **4.4 System Design**

The system design gives a description of the architectures, components, modules, interfaces and data for a specific system. It aids in studying how system components interact and function (Gemino & Parker, 2009). The sections below include notations used to describe the designed system.



#### 4.4.1 Context Diagram

A context diagram specifies details of the system design. It illustrates the external entities which interact with the system. It displays the inputs and outputs to and from the external entities. The figure below shows the context diagram of the web log analysis system.

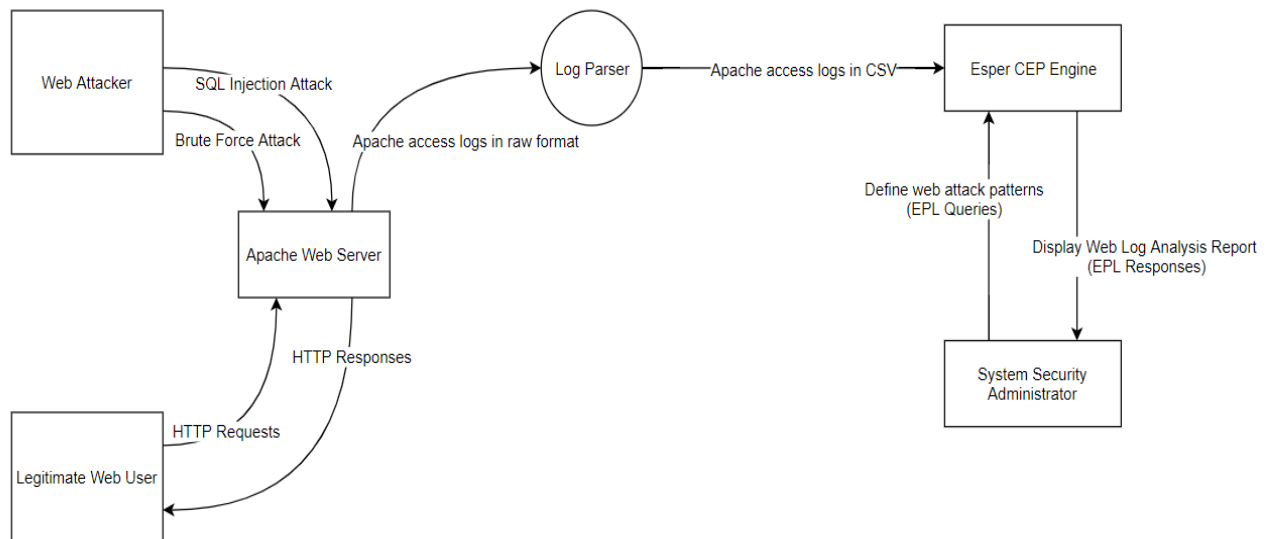


Figure 4. 4: Web Log Analysis Context Diagram

#### 4.5 System Model

The system model includes the system inputs, system processes and system outputs. The UML notation used in describing the system model includes the use case diagram and the sequence diagrams.

##### 4.5.1 Use Case Diagram

A use case diagram is a representation of the interactions between a user and a system or between a system and another system under observation hence capturing the functional aspects of the system (Aggarwal, 2005). Figure below gives an illustration of the use case for the web log analysis system.

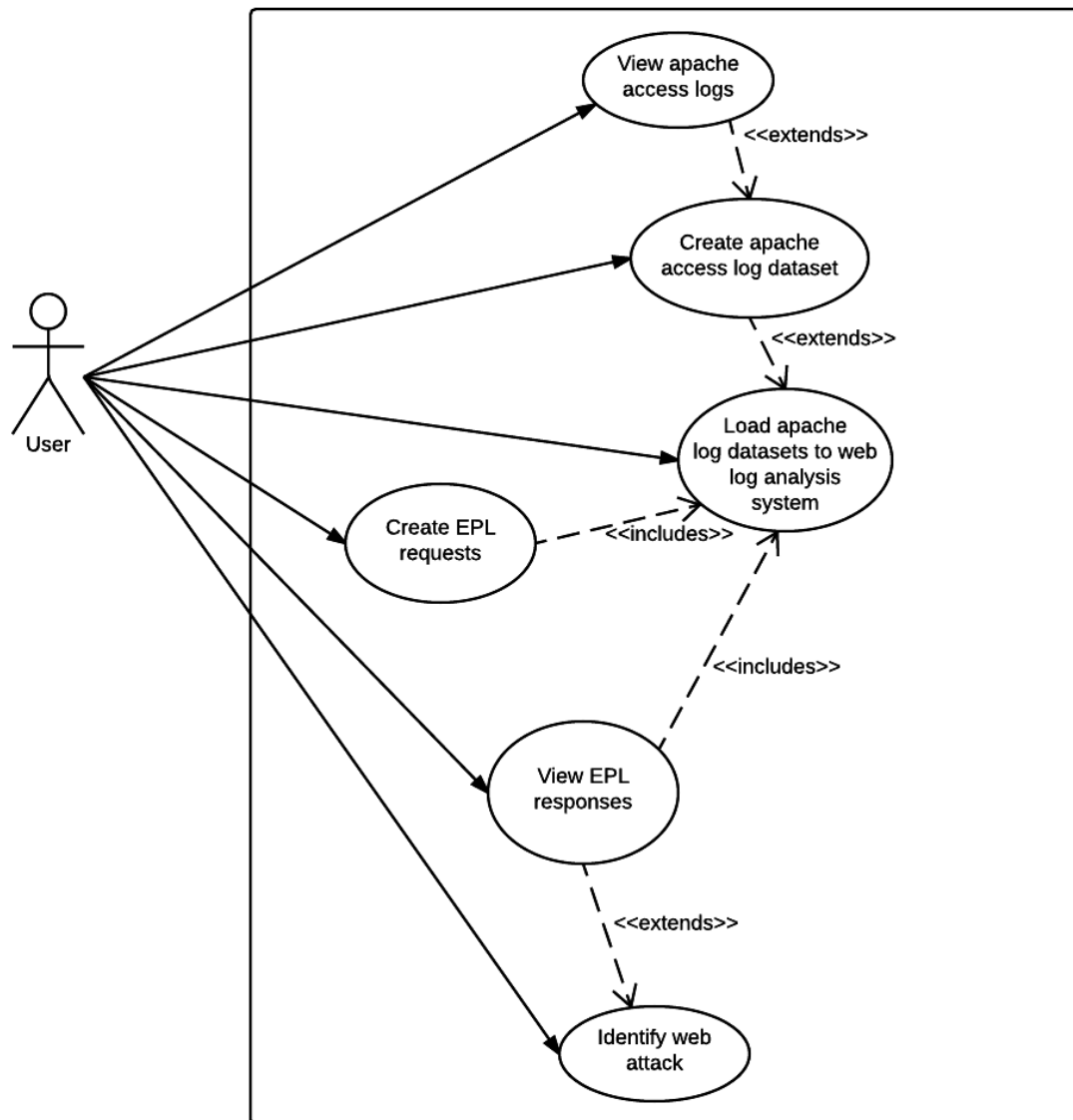


Figure 4. 5: Use Case Diagram for Web Server Log Analysis Platform

i. View Apache Access Logs Use Case Description

Use Case Title	View Apache Access Logs
Description	This use case scenario describes the steps of viewing Apache web access logs.
Actors	System Administrator (User)
Pre-Conditions	User has logged in as root.
Basic Flow	<ol style="list-style-type: none"> <li>1. User changes directory to /var /log/apache2/access.log</li> <li>2. The terminal displays access log metadata.</li> </ol>

Post Conditions	User is able to view the access logs and their metadata.
Frequency of Use	Many

Table 4. 2: View Apache Logs Use Case

ii. Create Apache Access Log Datasets Use Case Description

Use Case Title	Create Apache Access Log Datasets
Description	This use case scenario describes the steps of creating access log datasets.
Actors	System Administrator (User)
Pre-Conditions	User has logged in as root.
Basic Flow	<ol style="list-style-type: none"> <li>1. Create a log2CSV parser</li> <li>2. Run the parser across access logs</li> <li>3. Terminal displays access log metadata in CSV form.</li> </ol>
Post Conditions	User is able to convert access log metadata to CSV format.
Frequency of Use	Many

Table 4. 3: Create Apache Logs Dataset Use Case

iii. Load Datasets to Web Log Analysis System Use Case Description

Use Case Title	Create Apache Access Log Datasets
Description	This use case scenario describes the steps of loading the datasets to the web log analysis system
Actors	System Administrator (User)
Pre-Conditions	User accesses the Esper CEP engine.
Basic Flow	<ol style="list-style-type: none"> <li>1. Create a CSV Input Adapter java class.</li> <li>2. CVS Input Adapter reads log metadata from the CSV files and sends them to Esper as events.</li> </ol>

Post Conditions	User is able to convert access log metadata to CSV format.
Frequency of Use	Once

Table 4. 4: Load Datasets Use Case

iv. Create EPL Requests

Use Case Title	Create Apache Access Log Datasets
Description	This use case scenario describes the steps of loading the datasets to the web log analysis system
Actors	System Administrator (User)
Pre-Conditions	User accesses the Esper CEP engine.
Basic Flow	<ol style="list-style-type: none"> <li>1. User creates SQL like statements using Event Processing Language defining a specific web attacks.</li> <li>2. User runs these queries across log datasets.</li> </ol>
Post Conditions	User is able to create EPL queries.
Frequency of Use	Many

Table 4. 5: Create EPL Requests Use Case

v. View EPL Responses

Use Case Title	Create Apache Access Log Datasets
Description	This use case scenario describes the steps of loading the datasets to the web log analysis system
Actors	System Administrator (User)
Pre-Conditions	User accesses the Esper CEP engine.

Basic Flow	<ol style="list-style-type: none"> <li>1. User runs EPL queries across log datasets.</li> <li>2. System displays EPL results to user.</li> </ol>
Post Conditions	User is able to view EPL queries results.
Frequency of Use	Many

Table 4. 6: View EPL Responses Use Case

vi. Identify Web Attack

Use Case Title	Create Apache Access Log Datasets
Description	This use case scenario describes the steps of loading the datasets to the web log analysis system
Actors	System Administrator (User)
Pre-Conditions	User accesses the Esper CEP engine.
Basic Flow	<ol style="list-style-type: none"> <li>1. System displays EPL queries results to user.</li> <li>2. User identifies log metadata containing web attack evidence.</li> </ol>
Post Conditions	User is able to identify web attacks.
Frequency of Use	Many

Table 4. 7: Identify Web Attack Use Case

### 4.5.2 Sequence Diagrams

A sequence diagram displays how external system actors generate their order of intersystem events (Larman, 2002). Figures 4.6 and 4.7 below illustrate sequence diagrams indicating objects interaction.

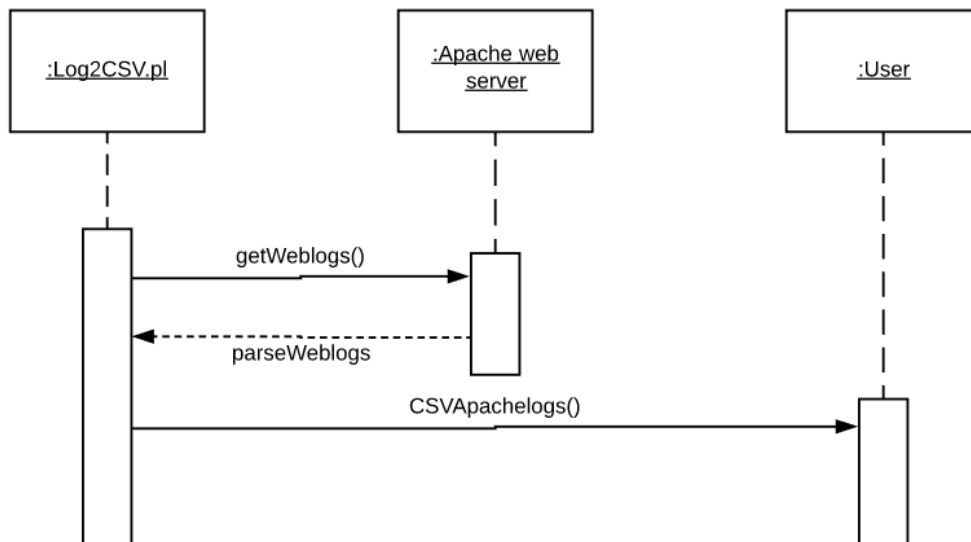


Figure 4. 6: Sequence Diagram for Data Collection

Figure 4.6 above represents the first stage in the system. The process is activated when the user runs the python script. Figure 4.7 below represents the second stage in the system, this include the steps taken for log analysis.

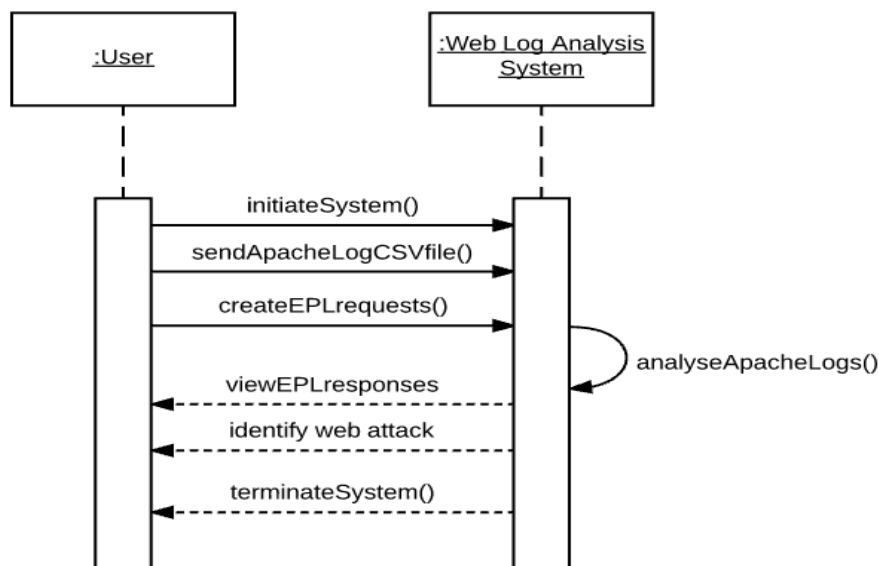


Figure 4. 7: Sequence Diagram for Log Analysis

Figure 4.7 above represents the second stage in the system. The process is activated when the user loads the Apache log CSV files to Esper CEP engine for processing. Esper then presents the results to user for analysis.

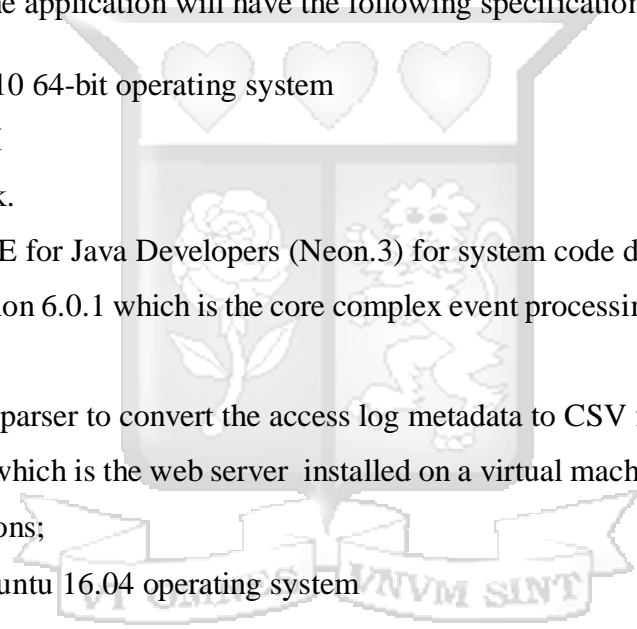
## CHAPTER FIVE: SYSTEM IMPLEMENTATION AND TESTING

### 5.1 Introduction

This chapter describes the implementation of the proposed system which involved Apache log parsing and Apache log analysis using temporal logic to detect HTTP attacks. The systems' key features are highlighted as well as system screenshots indicating users' screens and back end screens. This section of this chapter describes the various system tests carried out for its functionalities and usability according to the system objectives.

### 5.2 System Specification

The system holding the application will have the following specifications:

- 
- i. Windows 10 64-bit operating system
  - ii. 4GB RAM
  - iii. 20GB disk.
  - iv. Eclipse IDE for Java Developers (Neon.3) for system code development.
  - v. Esper version 6.0.1 which is the core complex event processing engine for log event analysis.
  - vi. Log2CSV parser to convert the access log metadata to CSV form.
  - vii. Apache 2 which is the web server installed on a virtual machine with the following specifications;
    - Ubuntu 16.04 operating system
    - Memory of 512 MB RAM
    - Hard disk size of 9 GB

### 5.3 System Implementation and Testing

#### 5.3.1 Web Server Configuration

Apache2 web server was installed in the Ubuntu virtual machine and a WordPress based web site was created and hosted on the Apache web server, see Appendix A. In order to capture the web server logs consisting of specific user activities, Apache2 service has to be started as shown below in figure 5.1.

```
peris@peris-VirtualBox:~$ sudo service apache2 start
[sudo] password for peris:
peris@peris-VirtualBox:~$
```

Figure 5. 1: Apache2 Start Service

The web server logs are stored in the access.log file under /var/log/apache2 directory.

```
peris@peris-VirtualBox:~$ cd /var/log/apache2
peris@peris-VirtualBox:/var/log/apache2$ ls
access.log      access.log.6.gz  error.log.13.gz  error.log.9.gz
```

Figure 5. 2: Web Server Logs Directory Location

### 5.3.1.1 CSV (Comma Separated Values)

CSV files were also used to store retrieved Apache access log metadata in comma separated values (CSV) form as in the example shown in figure 5.3 below.

```
"Host","Log Name","Date Time","Time Zone","Method","URL","Response Code","Bytes Sent","Referer","User Agent"
"192.168.56.1","-","2017-05-06 10:20:53","GMT+0300","GET","/",200,3525,"-","Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36"
"192.168.56.1","-","2017-05-06 10:20:53","GMT+0300","GET","/icons/ubuntu-logo.png",200,3623,"http://192.168.56.101/","
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36"
"192.168.56.1","-","2017-05-06 10:20:59","GMT+0300","GET","/favicon.ico",404,506,"http://192.168.56.101/","Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36"
```

Figure 5. 3: Web Server Logs CSV Format

### 5.3.2 Web Log Analysis Configuration

The main component of the log analysis, Esper, was installed in the Eclipse Neon.3 IDE running on Windows 10 operating system, see Appendix B. In order to reconstruct the web activities through Apache server log entries, the records from the access log file are first read then sent to the Esper engine as events. To do so, an Apache Log Input adapter was developed in Java for reading the access log metadata as input streams for processing.

#### 5.3.2.1 Apache Log Input Adapter Configuration

Currently Esper provides only 7 types of adapters such as AMQP, CSV and HTTP, none of them supports Apache log files hence an Apache Log input adapter had to be developed in Java language, see Appendix D. First the log records are converted to a CSV file via a log parser. The log parser, accesslog2csv.pl, written in Perl language was downloaded from GitHub website link <http://github.com/woonsan/accesslog2csv/blob/master/accesslog2csv.pl> (Last accessed 20th April 2018).



```

accesslog2csv.pl
if ("ARGV[0]" =~ /^-h|--help$/) {
    print "Usage: $0 access_log_file > csv_output_file.csv\n";
    print "    Or, $0 < access_log_file > csv_output_file.csv\n";
    print "    Or, $0 < access_log_file > csv_output_file.csv 2> invalid_lines.txt\n";
    exit(0);
}

%MONTHS = ( 'Jan' => '01', 'Feb' => '02', 'Mar' => '03', 'Apr' => '04', 'May' => '05', 'Jun' => '06',
    'Jul' => '07', 'Aug' => '08', 'Sep' => '09', 'Oct' => '10', 'Nov' => '11', 'Dec' => '12' );

print STDOUT "\"Host\\\", \"Log Name\\\", \"Date Time\\\", \"Time Zone\\\", \"Method\\\", \"URL\\\", \"Response Code\\\", \"Bytes Sent\\\", \"Referer\\\", \"User Agent\\\"\\n\"";
$line_no = 0;

while (<>) {
    ++$line_no;
    if (/^(["\w\.:~!@#$%^&*()-+=\[\]{}|;`'"/>
    $host = $1;
    $other = $2;
    $logname = $3;
    $day = $4;
    $month = $MONTHS{$5};
    $year = $6;
    $hour = $7;
    $min = $8;
    $sec = $9;
    $tz = $10;
    $method = $11;
    $url = $12;
    $code = $13;
    if ($14 eq '-') {
        $bytesd = 0;
    } else {
        $bytesd = $14;
    }
    $referer = $17;
    $ua = $18;

    print STDOUT "\"$host\\\", \"$logname\\\", \"$year-$month-$day $hour:$min:$sec\\\", \"$GMT$tz\\\", \"$method\\\", \"$url\\\", $code, $bytesd, \"$referer\\\", \"$ua\\\"\\n\"";
} else {
    print STDERR "Invalid Line at $line_no: $_";
}
}

```

Figure 5. 4: Log Parser Script

The input adapter is set to generate log events and read through the EPL queries. The code snippet below shows how the adapter is set up to retrieve log metadata from the log.java class, see Appendix C.

```

Configuration cepConfig = new Configuration();
cepConfig.addEventType("ApacheLog", Log.class.getName());
EPServiceProvider cep = EPServiceProviderManager.getProvider("ApacheInputAdapter", cepConfig);
EPRuntime cepRT = cep.getEPRuntime();

```

Figure 5. 5: Input Adapter Event Generation

The adapter loops over all EPL queries, this depends on the number of queries. For the adapter to loop over only one query, the system user can modify the for-loop in the code.

```

for(int queryCounter = 1; queryCounter <= 10; queryCounter++) {
    System.out.println("-----");
    System.out.println("PERFORMING QUERY " + queryCounter);
    System.out.println("-----");

    try {
        String query = new Scanner(new File("query " + queryCounter + ".epl")).useDelimiter("\\Z").next();
        cepAdm.destroyAllStatements();
        EPStatement cepStatement = cepAdm.createEPL(query);
        cepStatement.addListener(new CEPLListener());
    } catch (Exception e) {
        System.err.println("Error: " + e.getMessage());
    }
}

```

Figure 5. 6: Input Adapter for- loop

The adapter reads through the Apache logs in the CSV form and generates EPL responses based on the EPL queries.

```

try {
    File file = new File("Apacheaccesslogs.csv");
    BufferedReader reader = new BufferedReader(new FileReader(file));
    String line = null;
    StringTokenizer st = null;

    while((line = reader.readLine()) != null) {
        Log log = new Log("Apacheaccesslogs");

```

Figure 5. 7: Input Adapter CSV Configuration

See Appendix D for full configuration

Figure 5.8 below indicates the system network topology consisting of the system devices used.

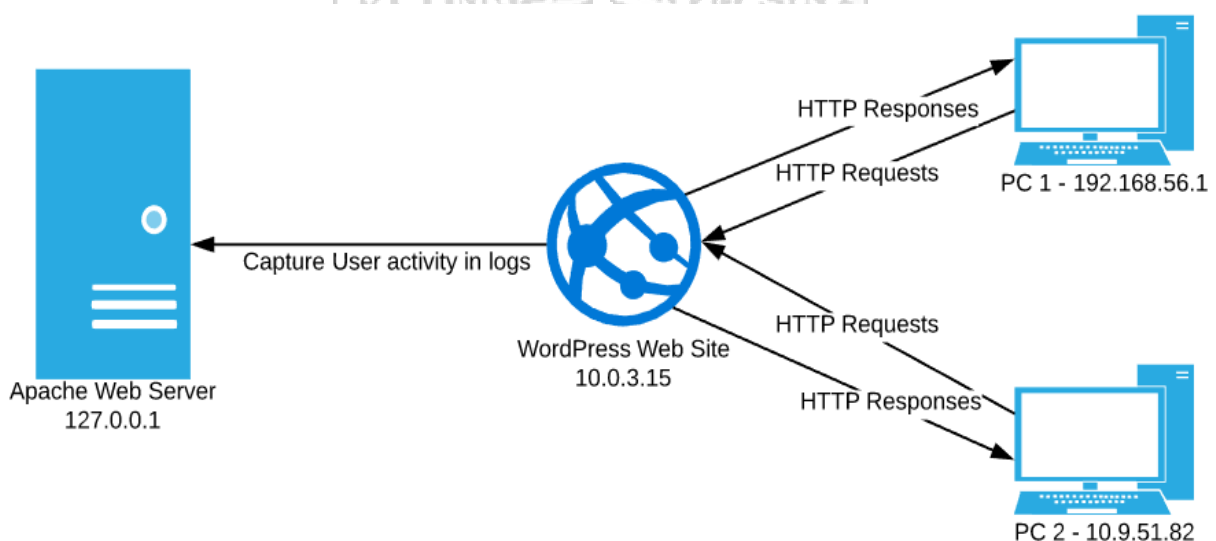


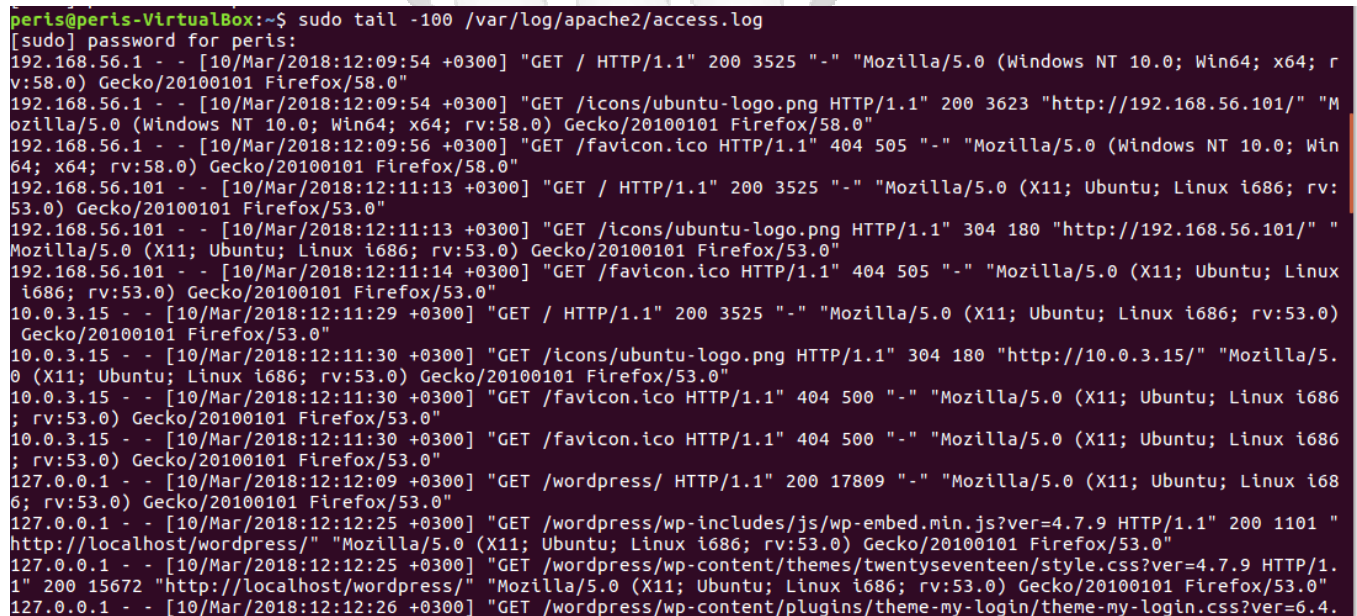
Figure 5. 8 : Web Log Analysis System Network Topology

## 5.4 System Features

Following the complete installation and configuration of all system components on the Ubuntu virtual and the Windows 10 machines, the following were the features available on the web log analysis platform.

### 5.4.1 Viewing Apache Access Logs

Specific system user activities on the web site are captured in the web server logs. In order to view the logs from the Apache 2 web server, the system user accesses the **var/log** server directory. Figure 5.9 shows an example of the access logs that were viewed by the system administrator using the command *sudo tail -100 /var/log/apache2/access.log* which displays the last 100 web server logs.



```
peris@peris-VirtualBox:~$ sudo tail -100 /var/log/apache2/access.log
[sudo] password for peris:
192.168.56.1 - - [10/Mar/2018:12:09:54 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.1 - - [10/Mar/2018:12:09:54 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 200 3623 "http://192.168.56.101/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.1 - - [10/Mar/2018:12:09:56 +0300] "GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.101 - - [10/Mar/2018:12:11:13 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
192.168.56.101 - - [10/Mar/2018:12:11:13 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 304 180 "http://192.168.56.101/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
192.168.56.101 - - [10/Mar/2018:12:11:14 +0300] "GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:29 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 304 180 "http://10.0.3.15/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:09 +0300] "GET /wordpress/ HTTP/1.1" 200 17809 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:25 +0300] "GET /wordpress/wp-includes/js/wp-embed.min.js?ver=4.7.9 HTTP/1.1" 200 1101 "http://localhost/wordpress/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:25 +0300] "GET /wordpress/wp-content/themes/twentyseventeen/style.css?ver=4.7.9 HTTP/1.1" 200 15672 "http://localhost/wordpress/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:26 +0300] "GET /wordpress/wp-content/plugins/theme-my-login/theme-my-login.css?ver=6.4.0 HTTP/1.1" 200 15672 "http://localhost/wordpress/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
```

Figure 5. 9: Apache Web Server Access Logs

### 5.4.2 Creating Web Server Logs Datasets

The web server access logs datasets were created through the process of log parsing. To do so, the Apache access log records were converted to CSV (Comma Separated Values) form. This was done through an Apache log parser written in perl script see figure 5.4 above.

Access log metadata included;

- i. Host
- ii. Log Name

- iii. Date Time
- iv. Time Zone
- v. Method
- vi. URL
- vii. Response Code
- viii. Bytes Sent
- ix. Referrer
- x. User Agent

Figure 5.10 below shows the raw format of the access logs.

```
192.168.56.101 - - [10/Mar/2018:12:11:14 +0300] "GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:29 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 304 180 "http://10.0.3.15/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
```

Figure 5. 10: Apache Logs in Raw Format

To convert the access log data to CSV, the command in figure 5.11 below was run and the records were saved in the accesslogs.csv file.

```
peris@peris-VirtualBox: ~
peris@peris-VirtualBox:~$ perl accesslog2csv.pl /var/log/apache2/access.log > 20180310accesslogs.csv
```

Figure 5. 11: Log Data Conversion to CSV Command

When parsed the raw access log format in figure 5.9 above can be displayed in a more detailed output with specified fields giving information on each metadata of the access log as seen in figure 5.12 below.

	A	B	C	D	E	F	G	H	I
1	Host	Date Time	Time Zone	Method	URL	Response Code	Bytes Sent	Referer	User
2	192.168.56.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-admin/	302	424	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
3	192.168.56.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/index.php/login/	200	19558	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
4	192.168.56.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/plug	200	1218	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
5	192.168.56.101	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/ther	200	2761	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
6	192.168.56.101	05/04/2018 14:49	GMT+0300	GET	/wordpress/wp-admin/profil	400	483	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
7	192.168.56.101	05/04/2018 14:52	GMT+0300	GET	/wordpress/wp-login.php?re	200	3494	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
8	10.9.51.82	05/04/2018 14:56	GMT+0300	POST	/wordpress/wp-login.php	302	1140	http://localhost/wordpress/wp-login	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
9	10.9.51.82	05/04/2018 14:56	GMT+0300	GET	/wordpress/wp-admin/profil	200	9419	http://localhost/wordpress/wp-login	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
10	10.9.51.82	05/04/2018 14:57	GMT+0300	GET	/wordpress/wp-admin/profil	400	483	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
11	127.0.0.1	05/04/2018 15:02	GMT+0300	GET	/wordpress/wp-login.php?re	200	3494	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
12	127.0.0.1	05/04/2018 15:02	GMT+0300	POST	/wordpress/wp-login.php	200	1881	http://localhost/wordpress/wp-login	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
13	127.0.0.1	05/04/2018 15:04	GMT+0300	POST	/wordpress/wp-login.php	302	1140	http://localhost/wordpress/wp-login	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
14	127.0.0.1	05/04/2018 15:04	GMT+0300	GET	/wordpress/wp-admin/profil	200	9427	http://localhost/wordpress/wp-login	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
15	127.0.0.1	05/04/2018 15:05	GMT+0300	POST	/wordpress/wp-admin/admin	200	436	http://localhost/wordpress/wp-admin	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
16	127.0.0.1	05/04/2018 15:06	GMT+0300	POST	/wordpress/wp-admin/admin	200	436	http://localhost/wordpress/wp-admin	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
17	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/ther	200	2958	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
18	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/ther	200	766	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
19	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/ther	200	15673	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
20	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-includes/js/w	200	4582	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
21	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/wordpress/wp-content/ther	304	182	http://localhost/wordpress/index.php	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0
22	127.0.0.1	05/04/2018 12:42	GMT+0300	GET	/favicon.ico	404	500	-	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0

Figure 5. 12: Apache Log CSV File

### 5.4.3 Loading Datasets to Web Log Analysis System

Following the creation of the Apache logs datasets in CSV format, the log files are then loaded to the CEP engine, Esper, for processing by the Apache Log Input adapter already created.

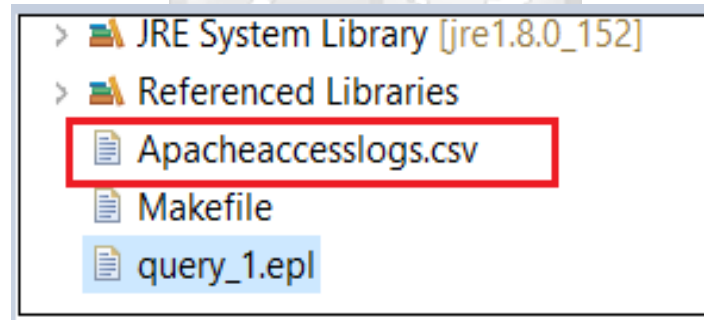


Figure 5. 13: Log CSV Loaded to Analysis System

### 5.4.4. Creating EPL Requests

The temporal formulas in MSFOMTL as indicated in figure 4.2 are translated to EPL (Event Processing Language) requests which are applied over Esper. Simple EPL requests are designed to filter the web log records one at a time over one input stream. This results in small memory space consumption. Fields, additional expressions or user- defined functions over those fields with key roles in introducing the activity to the system user in the SELECT clause. Filters are designed through WHERE clause where logical operators along with built-in or user

define functions are employed. Figures 5.13 and 5.14 shows SQL-like EPL queries defining SQL injection attack using GET method and POST method.

```
1 SELECT host, time, zone, url, response, user
2 FROM ApacheLogs
3 WHERE response = '400' and method = 'GET' and url like '%or%=%'
4
```

Figure 5. 14: SQL Injection Attack Using GET method EPL Query

The query above receives Apache log records as shown in line 2, filters out records of which response value is '400', method is equal to 'GET' and url contains the signature , '%or%=%' that most SQL injection attacks feature in line 3. . In line 1, output the filtered records as an intended activity that address the host, timestamp, time zone, url, response code and user agent containing the signature and agent used through the attack.

```
1 SELECT host, time, zone, user
2 FROM ApacheLogs
3 WHERE response = '200' and method = 'POST' and url like
4 "%$querypage%" and bytes > $threshold
5
```

Figure 5. 15: SQL Injection Attack Using POST method EPL Query

The query above receives Apache log records as shown in line 2, filters out records of which response value is '200', method is equal to 'POST', url value contains location of the dynamic query page (\$querypage) which has potential for SQL injection attack as shown in line 3. Since Apache log records do not store content through POST methods, one can define a maximum size value (\$threshold) for responses. In line 1, output the filtered records as an intended activity that address the host, timestamp, time zone , url and user agent containing the signature and agent used through the attack.

#### 5.4.5 Viewing EPL Responses

EPL requests created are run through the Apache logs with the aid of the Apache Log Input adapter. The EPL query defining a SQL injection attack is run through the Apacheaccesslog.CSV and the EPL response is as seen in figure 5.16 below.



```

PERFORMING QUERY 1
-----
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 14:52, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 14:56, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 15:02, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 15:04, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:42, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:43, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101
EVENT! {zone=GMT+0300, host=127.0.0.1, time=05/04/2018 12:43, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101

```

Figure 5. 16: EPL Response

### 5.4.6 Identifying Web Attacks

In order for the system administrator to identify a web attack through the log analysis, the following system steps are performed:

- i. The web attack is logged into the Apache log file

Traces of the SQL Injection attack using GET method are logged into the access.log file of the Apache web server as shown in the red box in figure 5.17 below.

```

127.0.0.1 - - [05/Apr/2018:14:56:49 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 302 1140 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:14:56:49 +0300] "GET /wordpress/wp-admin/profile.php HTTP/1.1" 200 9419 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:14:57:04 +0300] "GET /wordpress/wp-admin/profile.php%20or=% HTTP/1.1" 400 483 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"

```

Figure 5. 17: SQL Injection using GET Method Traces in Apache Log

- ii. Raw data log file records are converted to CSV and loaded to Esper

The raw data log files as shown in figure 5.17 above are then converted to CSV data format by running the command shown in figure 5.11. The CSV file is then loaded to the CEP engine, Esper, for processing as indicated in figure 5.13.

- iii. The CSV database is queried

The CSV database containing the log file records is then queried using temporal logic formula in Event Processing Language. The system administrator runs the SQL Injection attack using GET method EPL query, see figure 5.13.

#### iv. EPL result analysis

The EPL response is as shown in the figure 5.18 below is then analysed by the system administrator.

```
PERFORMING QUERY 1
EVENT! {method=GET, response=400, host=127.0.0.1, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0, url=/wordpress/wp-admin/profile.php%20%or%=%}
EVENT! {method=GET, response=400, host=127.0.0.1, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0, url=/wordpress/wp-admin/profile.php%20%or%=%}
```

Figure 5. 18: SQL Injection Attack using GET method Log Output

From the EPL query, the web log analysis system generates logs with response code equals to 400, method is equal to 'GET' and url contains the signature '%or%=%' as shown in the red box in figure 5.17 above.

## 5.5 System Testing

This section describes the various tests carried out on the system developed to ensure that it works well. The system was evaluated against both functional and non-functional requirements. System testing was categorized into two sections, developer testing and web system user testing. System tests performed by the developer were to ensure that the system's various functionalities were working well, tests included user acceptance testing and unit testing.

### 5.5.1 User Acceptance Testing

This testing was done by performing various web server user activities that were captured via the web server logs and were analysed by the web analysis platform. These activities were:

- i. Successful user login with administrator account.
- ii. Failed user login attempt.
- iii. Brute force attack.
- iv. SQL injection attack
- v. Identification of the attacks.

This proposed web server log analysis platform majorly focused on WordPress theme as the case study. The results for the above tests are displayed in the figures below, where these user activities were captured in the web logs then detected by the web server log analysis system. WordPress successful login is a web activity where the web user successfully logs into the web site as shown in figure 5.18 below.



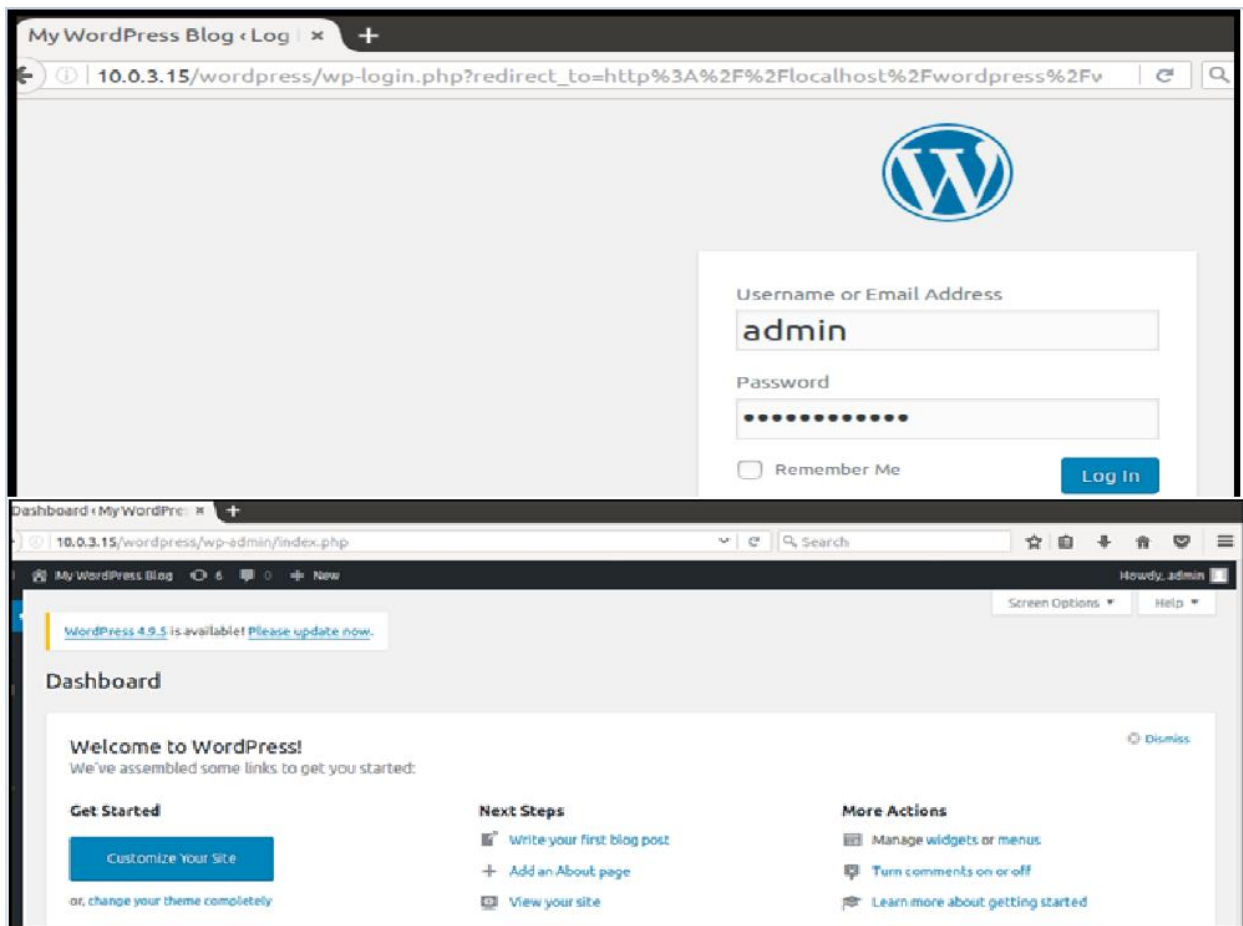


Figure 5. 19 : WordPress Successful Login

The website responds to the client with code 302 which corresponds to redirection of the page from wp-login.php. The green rectangle in figure 5.19 below indicates successful user login traces captured in the Apache logs.

```
127.0.0.1 - - [05/Apr/2018:12:49:38 +0300] "GET /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1 HTTP/1.1" 200 3494 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:12:49:46 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 200 1869 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:12:50:04 +0300] "GET /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1 HTTP/1.1" 200 3483 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:12:50:09 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 302 1307 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:12:50:09 +0300] "GET /wordpress/wp-admin/profile.php HTTP/1.1" 200 9423 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
peris@peris-VirtualBox:~$
```

Figure 5. 20: WordPress Login Traces in Apache Log

WordPress failed login is a web activity where the user attempts to log into the web site unsuccessfully as shown in figure 5.20 below.

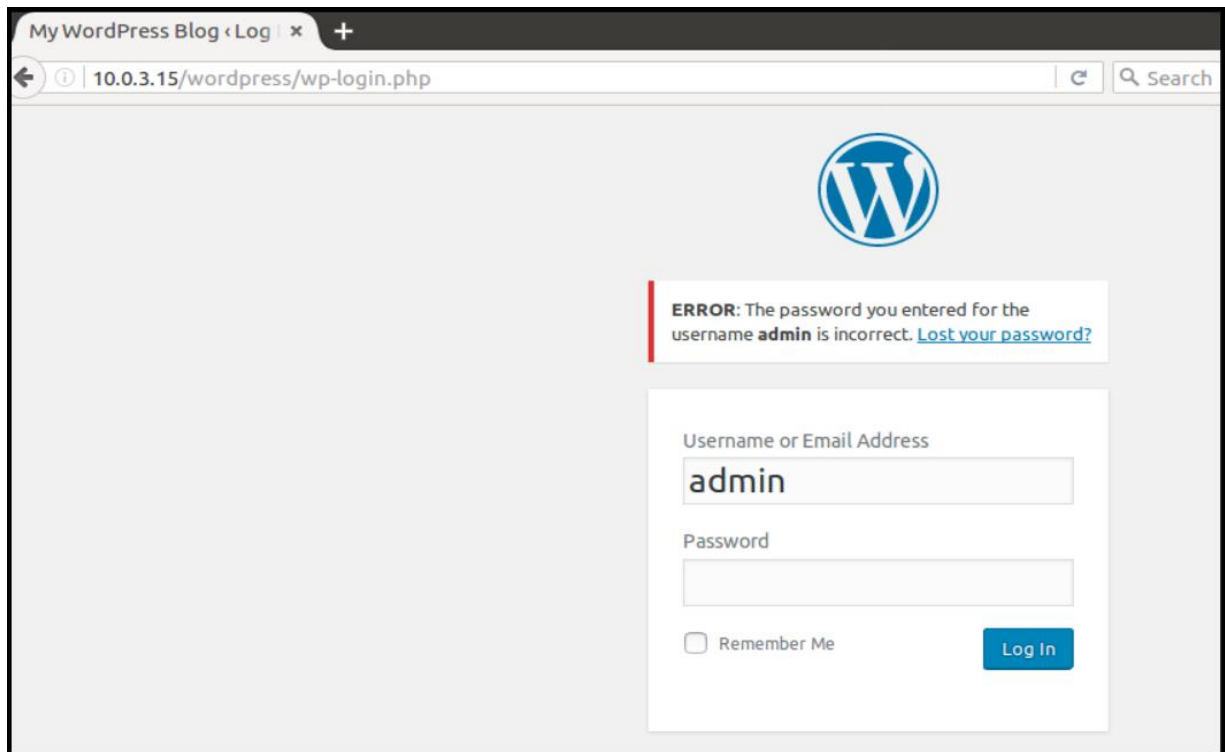


Figure 5. 21: WordPress Failed Login

WordPress login page, wp-login.php is designed to accept login credentials through POST method. The website responds with code 200. This means the page returns successfully with a response message indicating a login failure.

```
127.0.0.1 - - [05/Apr/2018:15:02:03 +0300] "GET /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1 HTTP/1.1" 200 3494 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:15:02:09 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 200 1881 "http://localhost/wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
```

Figure 5. 22: WordPress Failed Login Traces in Apache Logs

In web server misuse cases where the web user tries to retrieve the login credentials through a trial and error method.

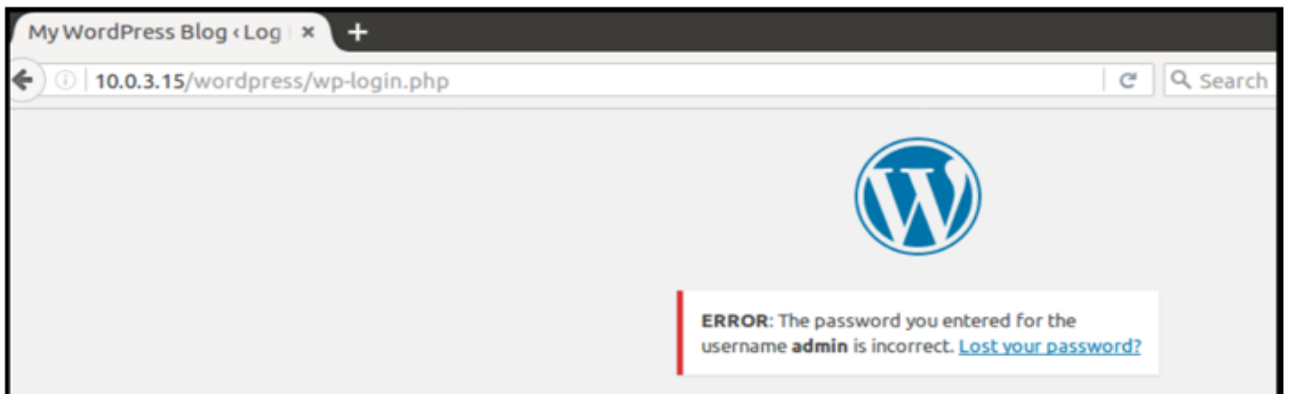


Figure 5. 23 : Failed Login Attempt One

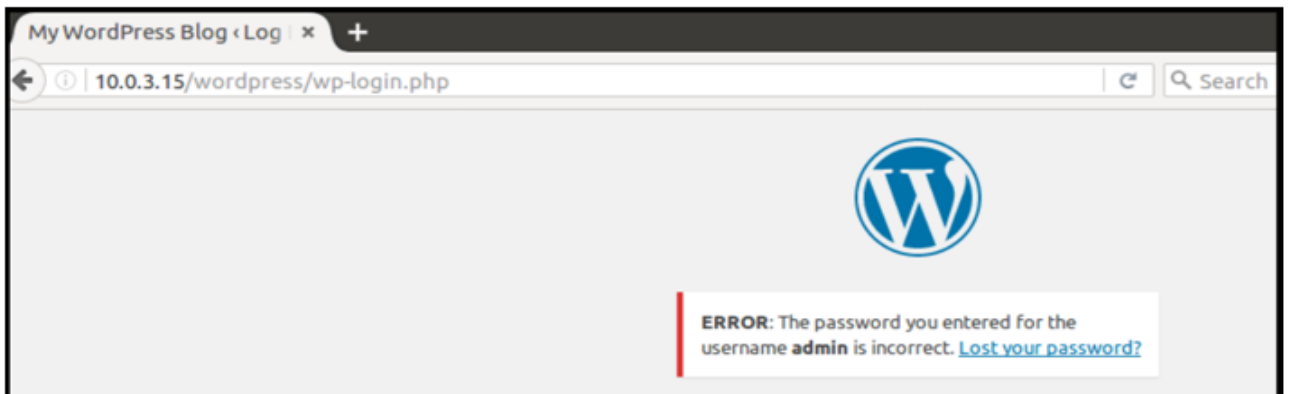


Figure 5. 24 : Failed Login Attempt Two

The continual failed login attempts traces are captured in the Apache logs as web activities entailing a repeating pattern.

```
127.0.0.1 - - [05/Apr/2018:14:36:32 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 200 1881 "http://localhost/wordpress/wp-login.php" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:14:36:40 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 200 1881 "http://localhost/wordpress/wp-login.php" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:14:36:47 +0300] "POST /wordpress/wp-login.php HTTP/1.1" 200 1881 "http://localhost/wordpress/wp-login.php" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [05/Apr/2018:14:38:57 +0300] "GET /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.php&reauth=1 HTTP/1.1" 200 3483 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
```

Figure 5. 25: Brute Force Login Attack Traces in Apache Logs

The system administrator then proceeds to creating EPL query emulating the attack as shown in figure 5.25 below.

```
1 SELECT host, time, zone, url, response, user
2 FROM ApacheLogs
3 WHERE response = '200' and method = 'POST'
4
```

Figure 5. 26 : Brute Force EPL Query

The query above receives Apache log records as shown in line 2, filters out records of which response value is '200' and method is equal to 'POST' that most brute force attacks feature in line 3.. In line 1, output the filtered records as an intended activity that address the host, timestamp, time zone, url, response code and user agent containing the signature and agent used through the attack

```

PERFORMING QUERY 1
-----
EVENT! {method=POST, response=200, host=127.0.0.1, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0, url=/wordpress/wp-login.php}
EVENT! {method=POST, response=200, host=127.0.0.1, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0, url=/wordpress/wp-admin/admin-ajax.php}
EVENT! {method=POST, response=200, host=127.0.0.1, user=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0, url=/wordpress/wp-admin/admin-ajax.php}
EVENT! {method=POST, response=200, host=127.0.0.1, user=WordPress/4.7.10; http://localhost/wordpress, url=/wordpress/wp-admin/upgrade.php?step=upgrade_db}
EVENT! {method=POST, response=200, host=127.0.0.1, user=WordPress/4.7.9; http://localhost/wordpress, url=/wordpress/wp-cron.php?doing_wp_cron=1522921357.4530150890350341796875}

```

Figure 5. 27 : Brute Force Log Output

## 5.5.2 Unit and Integration Testing

In unit testing, the specific system units were tested for operation. The software components was tested separately to ensure that each performed their required function. Integration testing was performed when two or more system components were integrated and tested for their system functionality. During unit and integration testing, some system errors were encountered and rectified. Tests were done by checking if each system component were displaying the required output.

### 5.5.2.1 Apache2 Web server

Apache2 web server was installed in Ubuntu 16.04 environment. A WordPress web site was designed and hosted in the web server. The test was to check if the Apache web server activities were being captured .This was done by first starting the Apache service and checking the logs in the access.log file.

```

peris@peris-VirtualBox:~$ sudo tail -100 /var/log/apache2/access.log
[sudo] password for peris:
192.168.56.1 - - [10/Mar/2018:12:09:54 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.1 - - [10/Mar/2018:12:09:54 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 200 3623 "http://192.168.56.101/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.1 - - [10/Mar/2018:12:09:56 +0300] "GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0"
192.168.56.101 - - [10/Mar/2018:12:11:13 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
192.168.56.101 - - [10/Mar/2018:12:11:13 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 304 180 "http://192.168.56.101/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
192.168.56.101 - - [10/Mar/2018:12:11:14 +0300] "GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:29 +0300] "GET / HTTP/1.1" 200 3525 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /icons/ubuntu-logo.png HTTP/1.1" 304 180 "http://10.0.3.15/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
10.0.3.15 - - [10/Mar/2018:12:11:30 +0300] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:09 +0300] "GET /wordpress/ HTTP/1.1" 200 17809 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:25 +0300] "GET /wordpress/wp-includes/js/wp-embed.min.js?ver=4.7.9 HTTP/1.1" 200 1101 "http://localhost/wordpress/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:25 +0300] "GET /wordpress/wp-content/themes/twentyseventeen/style.css?ver=4.7.9 HTTP/1.1" 200 15672 "http://localhost/wordpress/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:53.0) Gecko/20100101 Firefox/53.0"
127.0.0.1 - - [10/Mar/2018:12:12:26 +0300] "GET /wordpress/wp-content/plugins/theme-my-login/theme-my-login.css?ver=6.4.

```

Figure 5. 28: Web Server Activities Captured in Apache Logs

Figure 5.20 above shows us that the web server activities were being captured hence the Apache service was running properly.

### 5.5.2.2 Esper Complex Processing Engine

The Esper CEP was configured to run through the log files with the aid of the Apache Log Input Adapter and EPL queries. The figure 5.21 below shows the engine's output once queried to select all log file events.

Host: Host	Time: Time	Zone: Zone	Method: Method	Url: URL	Response: Response	Bytes: Bytes	Referer: Referer	Use
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-admin/	Response: 302	Bytes: 424	Referer: -	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/index.php/login/?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2F&reau			Referer: -	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/plugins/theme-my-login/theme-my-login.css?ver=6.4.9	Response: 400	Bytes: 483	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/themes/twentyseventeen/assets/js/jquery.scrollTo.js?ver=2.1.2	Response: 400	Bytes: 483	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 14:49	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-admin/profile.php%20%or%-%	Response: 400	Bytes: 483	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 14:52	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.p	Response: 302	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 14:56	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-login.php	Response: 302	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 14:56	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-admin/profile.php	Response: 200	Bytes: 9419	Referer: Ref	
Host: 127.0.0.1	Time: 05/04/2018 14:57	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-admin/profile.php%20%or%-%	Response: 400	Bytes: 483	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 15:02	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-login.php?redirect_to=http%3A%2F%2Flocalhost%2Fwordpress%2Fwp-admin%2Fprofile.p	Response: 200	Bytes: 1881	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 15:02	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-login.php	Response: 200	Bytes: 1881	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 15:04	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-login.php	Response: 302	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 15:04	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-admin/profile.php	Response: 200	Bytes: 9427	Referer: Ref	
Host: 127.0.0.1	Time: 05/04/2018 15:05	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-admin/admin-ajax.php	Response: 200	Bytes: 436	Referer: Ref	
Host: 127.0.0.1	Time: 05/04/2018 15:06	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-admin/admin-ajax.php	Response: 200	Bytes: 436	Referer: Ref	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/themes/twentyseventeen/assets/js/global.js?ver=1.0	Response: 200	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/themes/twentyseventeen/assets/js/skip-link-focus-fix.js?ver=1.0	Response: 200	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/themes/twentyseventeen/style.css?ver=4.7.9	Response: 200	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-includes/js/wp-emoji-release.min.js?ver=4.7.9	Response: 200	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /wordpress/wp-content/themes/twentyseventeen/assets/images/header.jpg	Response: 200	Bytes: 1140	Referer: ht	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /favicon.ico	Response: 404	Bytes: 500	Referer: -	
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: GET	Url: /favicon.ico	Response: 404	Bytes: 500	Referer: -	
Host: ::1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: OPTIONS	Url: *	Response: 200	Bytes: 126	Referer: -	User: Apache
Host: ::1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: OPTIONS	Url: *	Response: 200	Bytes: 126	Referer: -	User: Apache
Host: ::1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: OPTIONS	Url: *	Response: 200	Bytes: 126	Referer: -	User: Apache
Host: 127.0.0.1	Time: 05/04/2018 12:42	Zone: GMT+0300	Method: POST	Url: /wordpress/wp-admin/upgrade.php?step=upgrade_db	Response: 200	Bytes: 260	Referer: -	

Figure 5. 29: All Log Events Output



## CHAPTER SIX: DISCUSSION OF RESULTS

### 6.1 Introduction

The purpose of the dissertation was to study, understand and examine typical attacks on web servers focusing on HTTP attacks, to identify and generate log files of these attacks for analysis, to review the existing systems available for web server log analysis, to design and develop a web server log analysis platform that can be used to define attacks patterns using temporal logic and to validate the capability of the proposed solution in increasing the quality of log analysis to detect web attacks. This was done in order to develop a suitable technique that will be used by system and network administrators to monitor and identify web server misuse activities.

### 6.2 Findings and Achievements

A review of the literature indicated that there is an increased rate of web server misuse. Web servers' events are stored in the log files which server as important sources of evidence during web server misuse or crime incidents. The existing systems used for the extraction and analysis of these evidence face major challenges due to the enormous sizes of web log files and complexities in understanding the attack patterns connected to the crime. This leads to slow log analysis which is time consuming. Currently, system network and security professional lack efficient standard platforms to define and share attack patterns in regards to log analysis.

The developed system focused on web server misuse and analysis of Apache log files. Formal attack patterns were developed in MSFOMTL (Many Sorted First Order Metric Temporal Logic) and the related queries in EPL. The analysis platform is built on Esper, and tests carried out reveal that the system increases the quality of log analysis process and would aid in digital forensics aspects to identify system misuse cases in a timely manner. 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. EPL queries can be saved in EPL libraries, this in turn speeds up misuse investigations as it saves time by focusing on something novel instead of searching through already know attack patterns.

### 6.3 Review of Research Objectives

This dissertation identifies the challenges faced by system and network administrators in identifying web server misuses cases. A web server log analysis was developed with a selected technique from the literature review and the system results from the system analysis. This research was guided by the five research objectives outlined in Chapter 1.

The first objective was to study, understand and examine typical attacks on web servers focusing on HTTP; to the extent of this objective, these include cross site scripting, SQL injection, file inclusion and brute force attacks. The second objective was to identify and generate log files of these attacks for analysis; the main log file focused in this research was the access log file where traces of these web attacks were identified.

The third objective was to review the existing systems available for web server log analysis; a number of web log analysis applications and mechanisms were identified and their merits and limitations were reviewed and documented. The forth objective was to develop a web server log analysis system used to define web attack patterns using temporal logic; attack patterns were defined through MSFOMTL while corresponding queries were written in EPL. The EPL queries were run on Esper, an open source CEP engine, resulting to identification of web attacks. The fifth and final objective was to test the system developed to proof its capability in increasing the quality of log analysis to detect web attacks; through system testing and evaluation, the expected results were verified in accordance to the system functional requirements.

## CHAPTER SEVEN: CONCLUSIONS AND RECOMMENDATIONS

### 7.1 Introduction

This chapter gives a summary of the research study. The various existing methods and web log analysis techniques researched and implemented by different authors were reviewed and technological gaps and options available for this system were identified. The design, architecture and requirements of this system were identified as well. Temporal logic approach was used for system implementation. System testing and evaluation was performed throughout system development.

### 7.2 Conclusions

The sole purpose of this research was to develop a system that would aid network and system security administrators identify HTTP attacks, specifically SQL injection and brute force attacks through analysis of web server logs using temporal logic approach and log reconstruction. This comes as an aid for business and IT entities to ensure that they are able to identify web server misuses. The system tests revealed this analysis platform increases the quality of log analysis process thus contributing to digital forensics in various aspects. This approach could be used to enhance host based intrusion detection mechanisms.

### 7.3 Recommendations

From the results discovered during the research study, the following came out as recommendations;

- i. The system should be developed to analyse more web attacks such as cross site scripting, file inclusion, Xpath injection and command execution detection. This gives the system more flexibility.
- ii. The system accuracy can be enhanced by having more functionalities that focus on analysing logs located in the error.log files.
- iii. The system should be made available for other web themes including Joomla and Drupal.
- iv. The system should be developed to analyse other log files such as system security and audit events not just web server events.



## 7.4 Future Work

With further research, this system may be applicable in the business intelligence sector in use case analysis. In addition, it may aid in enhancing SIEM and intrusion detection mechanisms in building misuse and anomaly based Network Intrusion Detection Systems in which temporal formalisms for representing attack patterns are combined. Network and System Security administrators will then be able to run a number of queries previously stored in the EPL libraries. This will greatly save on time during web server attack investigations as the focus will be on searching for new attacks rather than searching for attack patterns already known and stored.



## REFERENCES

- Ahmed, A., Lisitsa, A., & Dixon, C. (2011). *A misuse-based network intrusion detection system using temporal logic and stream processing*. Network and System Security International Conference.
- Albek, E., Bax, E., Billock, G., Chandy, K. M., & Swett, I. (2005). *An event processing language (epl) for building sense and respond applications*. IEEE International Parallel and Distributed Processing Symposium.
- Arasteh, R. A., Debbabi, M., Sakha, A., & Saleh, M. (2007). *Analyzing multiple logs for forensic evidence*. Digital Investigation.
- Aulds, C. (2000). *Linux Apache Web Server Administration (Craig Hunt Linux Library Series)*. SYBEX Inc.
- Barret, B. (2014, May 28). Retrieved January 7, 2017, from Home of the Webalizer: <http://www.webalizer.org/>
- Borba, P., & Cavalcanti, A. (2007). *Testing Techniques in Software Engineering*. Brazil: PSSE.
- Bryman, J. M., & Bell, E. (2007). *Business Research Methods Revised Edition*. Oxford University Press.
- Calyptix. (2016, August 1). *Top 5 Cyber Attacks Types in 2016*. Retrieved January 4, 2017, from <http://www.calyptix.com/top-threats/top-5-cyber-attack-types-in-2016-so-far/>
- Calzarossa, M. C., & Massari, L. (2011). *Analysis of web logs: challenges and findings*. Springer Berlin Heidelberg.
- Cohen, M. I. (2008). *PyFlag—An advanced network forensic framework*. Digital investigation.
- Destailleur, L. (2015). Retrieved January 7, 2017, from AWStats log file analyzer: <http://www.awstats.org/>
- Fry, A. (2011). *A Forensic web Log Analysis Tool: Techniques and implementation*. Québec, Canada: Concordia University Montréal.
- Gemino, A., & Parker, D. (2009). *Use case diagrams in support of use case modeling: Deriving understanding from the picture*. Journal of Database Management.
- Goel, N., & Jha, K. C. (2013). *Analyzing Users' Behavior from Web Access Logs using Automated Log Analyzer tool*. International Journal of Computer Applications.
- Gordey, S. (2010). *Web Application Security Statistics*. Retrieved January 4, 2017, from <http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics>

- Grace, J. L., Maheswari, V., & Nagamalai, D. (2011). *Analysis of Web Logs and Web User in Web Mining*. International Journal of Network Security & Its Applications.
- Gunestas, M., & Bilgin, Z. (2016). *Log Analysis Using Temporal Logic and Reconstruction Approach: Web Server Case*. The Journal of Digital Forensics, Security and Law.
- Halfond, W. G., Viegas, J., & Orso, A. (2006). *A classification of SQL-injection attacks and countermeasures*. International Symposium on Secure Software Engineering .
- Harvin, H. (2016, August 27). *Agile Estimation and Planning*. Retrieved from Henry Harvin Education: <http://certificationcourses.henryharvin.com/Agile-development-methodology-provides-opportunities-to-assess-the-direction-of-a-project-throughout-the-development-lifecycle-Agile-methodologies-are-an-alternative-to-waterfall-or-traditional-seq/b94>
- Haven, R. W., Lunt, B., & Teng, C. C. (2012). *Naïve Bayesian filters for log file analysis: De-spam your logs*. In 2012 IEEE Network Operations and Management Symposium.
- Herreman, D. (2006). *EsperTech Event Series Intelligence*. Retrieved December 17, 2016, from <http://www.espertech.com/esper/>
- Herrerías, J., & Gómez, R. (2010). *Log analysis towards an automated forensic diagnosis system*. Availability, Reliability, and Security International Conference.
- Huth, M., & Ryan, M. (2004). *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press.
- Irny, S. I., & Rose, A. A. (2005). *Designing a Strategic Information Systems Planning Methodology for Malaysian Institutes of Higher Learning*. Issues in Information Systems , VI (1).
- Jain, K. R., Kasana, R. S., & Jain, S. (2009). *Efficient Web Log Mining using Doubly Linked Tree*. International Journal of Computer Science and Information Security.
- Janot, E., & Zavarsky, P. (2008). *Preventing SQL Injections in Online Applications*. Ghent, Belgium: Application Security Conference.
- Jayathilake, P. W. (2011). *A novel mind map based approach for log data extraction*. International Conference on Industrial and Information Systems .
- Jerkovic, J. I. (2009). *Essential Techniques for Increasing Web Visibility SEO warrior*. O'Reilly Media, Inc. Retrieved from [http://yourproseo.com/wp-content/uploads/2014/10/seo\\_warrior.pdf](http://yourproseo.com/wp-content/uploads/2014/10/seo_warrior.pdf)
- Kalamatianos, T., Kontogiannis, K., & Matthews, P. (2012). *Domain independent event analysis for log data reduction*. In 2012 IEEE 36th Annual Computer Software and Applications Conference.

- Kothari, C. (2004). *Research Methodology*. New Age International.
- Koymans, R. (1990). Specifying real-time properties with metric temporal logic. Real-time systems.
- Kumar, C. (2016). *Apache Web Server Hardening & Security Guide*. Retrieved February 2, 2017, from <https://geekflare.com/apache-web-server-hardening-security/>
- Levy, Y., & Ellis, T. J. (n.d.). 2011.
- Levy, Y., & Ellis, T. J. (2011). *A guide for novice researchers on experimental and quasi-experimental studies in information systems research*. Interdisciplinary Journal of information, knowledge, and management.
- Makanju, A., Nur Zincir-Heywood, A., & Milios, E. E. (2009). *Clustering event logs using iterative partitioning*. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.
- 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.
- Oliver, B. (2010). *Teaching Fellowship: Benchmarking Partnerships for Graduate Employability*. Cutin University.
- OWASP. (2016). Retrieved February 2, 2017, from Command Injection: [https://www.owasp.org/index.php/Command\\_Injection](https://www.owasp.org/index.php/Command_Injection)
- OWASP Application Security Community. (n.d.). Retrieved December 15, 2016, from Welcome to OWASP: [https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page)
- PyFlag. (n.d.). Retrieved January 7, 2017, from PyFlag Tutorial: <http://pyflag.sourceforge.net/Documentation/tutorials/>
- Ristic, I. (2005). *Apache Security*. Retrieved December 14, 2016, from The complete guide to securing your Apache Server: <https://www.feistyduck.com/library/apache-security/online/index.html>
- Roll- Hansen, N. (2009). *Why the distinction between basic (theoretical) and applied (practical) reserch is important in the politics of science*. London: Centre of Philosophy of Natural and Socila Science Contingency and Dissent in Science.
- Sawmill. (2010). Retrieved January 7, 2017, from Analyze, Monitor, Alert. Sawmill, Universal Log File Analysis and Reporting: <https://www.sawmill.net/index.html>

- Sharma, A. K., & Gupta, P. C. (2013). *Analysis of Web Server Log Files to Increase the Effectiveness of the Website Using Web Mining Tool*. International Journal of Advanced Computer and Mathematical Sciences.
- Singer, A., & Bird, T. (2004). *Building a Logging Infrastructure*. The Usenix Association.
- Sowmya, G., & Kumar, A. N. (2013). *Brute Force Attack - Blocking Techniques*.
- Srivastava, J., Cooley, R., Deshpande, M., & Tan, P.-N. (2000). *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data*. ACM Sigkdd Explorations Newsletter.
- Stockley, M. (2016, June 15). *Naked Security*. Retrieved February 8, 2017, from The web attacks that refuse to die: <https://nakedsecurity.sophos.com/2016/06/15/the-web-attacks-that-refuse-to-die/>
- Suneetha, K. R., & Krishnamoorthi, D. R. (2009). *Identifying user behavior by analyzing web server access log file*. IJCSNS International Journal of Computer Science and Network Security.
- Szalvay, V. (2004). *An Introduction to SA Agile Software Development*. Danube Technologies.
- Tutorials Point. (2006). Retrieved from SDLC Agile Model: [https://www.tutorialspoint.com/sdlc/sdlc\\_agile\\_model.htm](https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm)
- Tyagi, A., & Choudhary, S. (2015). *Web Usage Mining Using Web Log Expert Tool*. International Journal of Advanced Research in Computer Science and Software Engineering.
- Vernekar, S. S., & Buchade, A. (2013). *Map-Reduce based log file analysis for system threats and problem identification*. In Advance Computing Conference (IACC), 2013 IEEE 3rd International.
- Vianello, V., Gulisano, V., Jimenez-Peris, R., Patiño-Martínez, M., Torres, R., Díaz, R., & Prieto, E. (2013). *A Scalable SIEM correlation engine and its application to the Olympic games IT infrastructure*. Availability, Reliability and Security International Conference.
- Wainwright, P. (2008). *Pro Apache* (Third ed.). Appres.
- Web Technologies. (2016). Retrieved February 2, 2017, from Usage of web servers for websites: [https://w3techs.com/technologies/overview/web\\_server/all](https://w3techs.com/technologies/overview/web_server/all)
- Web Technology Survey. (2016). Retrieved January 2, 2017, from Usage of content management systems for websites: [https://w3techs.com/technologies/overview/content\\_management/all](https://w3techs.com/technologies/overview/content_management/all)

*WebLog Expert*. (n.d.). Retrieved January 7, 2017, from Powerful log analyzer:  
<http://www.weblogexpert.com/>

(2014, May 28). Retrieved January 7, 2017, from Home of the Webalizer:  
<http://www.webalizer.org/>

## APPENDICES

### Appendix A: Apache2 Web Server Installation

Pre- requisites- Ubuntu 16.04 operating system with 512 MB RAM and Hard disk size of 9 GB

The local package index has to be updated to reflect the latest upstream changes. Afterward Apache was installed using Ubuntu's package manager, apt, as shown in figure below.

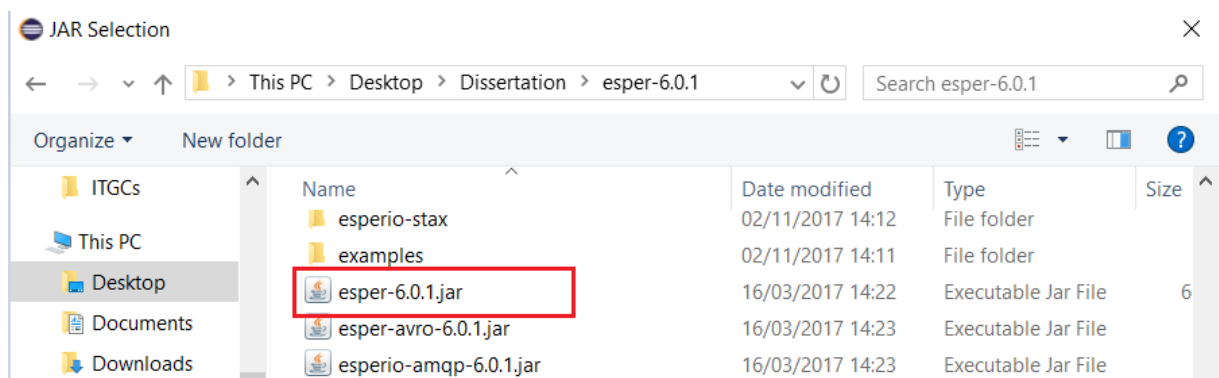
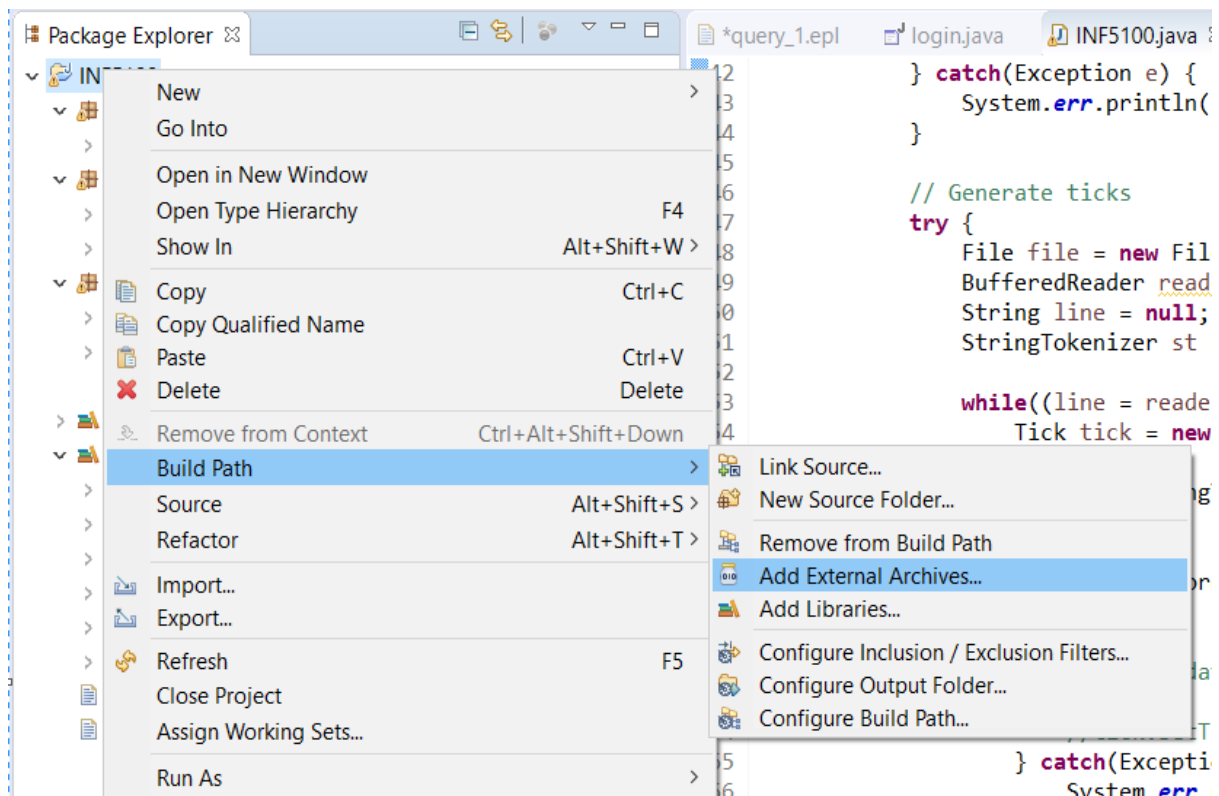
```
peris@peris-VirtualBox:~$ sudo apt-get install apache2
```

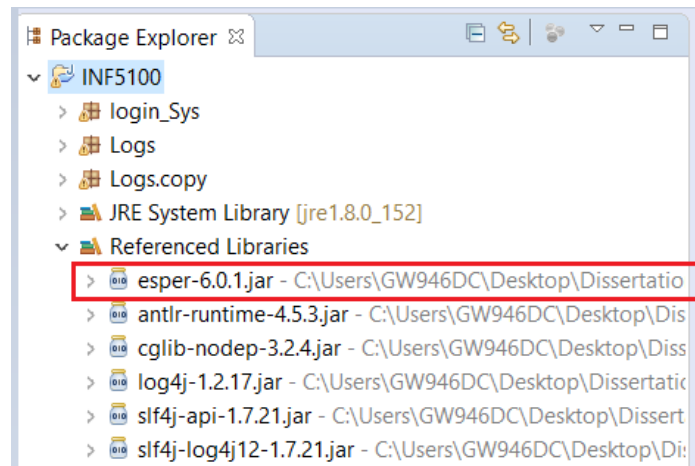
To confirm Apache installation, start the Apache service and direct web browser in use to <http://192.168.56.101, 10.0.3.15>, and the local host, to view Apache2 default web page.



## Appendix B: Esper Configuration in Eclipse IDE

To add Esper to the project, right click on the project name select Build path -> Add External Archives. Browse to the esper- 6.0.1.jar file.





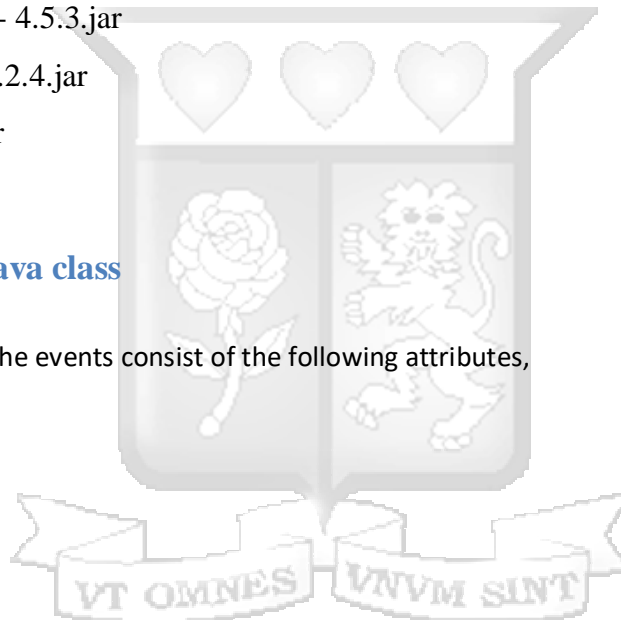
In order to run the project, add the following external archives

- Antlr- runtime- 4.5.3.jar
- Cglib-nodep-3.2.4.jar
- Log4j-1.2.7.jar

## Appendix C: Log.java class

This is the event class. The events consist of the following attributes,

- Host
- Time
- Zone
- Method
- Url
- Response
- Bytes
- Referer
- User



**Package** Logs;

```
public class Log {
    Private String      symbol;

    Private String      host;           //Host
    Private String      time;           //DateTime
    Private String      zone;           //TimeZone
    Private String      method;         //Method
    Private String      url;            //URL
    Private String      response;        ResponseCode
    Private String      bytes;          //Bytes Sent
    Private String      referer;        //Referer
    Private String      user;           //UserAgent
}
```



```

    public Log (String symbol) {
        this.symbol = symbol;
    }

    public Log(String symbol, String host, String time, String zone, String
method, String url, String response, String bytes, String referer, String user) {
        this.symbol = symbol;
        this.host = host;
        this.time= time;
        this.zone = zone;
        this.method = method;
        this.url = url;
        this.response = response;
        this.bytes = bytes;
        this.referer = referer;
        this.user =user;
    }

    public String getSymbol() {
        return symbol;
    }

    public void setSymbol(String symbol) {
        this.symbol = symbol;
    }

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getZone() {
        return zone;
    }

    public void setZone(String zone) {
        this.zone = zone;
    }

    public String getMethod() {
        return method;
    }

    public void setMethod(String method) {
        this.method = method;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getResponse() {
        return response;
    }

```

```

    }
    public void setResponse(String response) {
        this.response = response;
    }
    public String getBytes() {
        return bytes;
    }
    public void setBytes(String bytes) {
        this.bytes = bytes;
    }
    public String getReferer() {
        return referer;
    }
    public void setReferer(String referer) {
        this.referer = referer;
    }
    public String getUser() {
        return user;
    }
    public void setUser(String user) {
        this.user = user;
    }

    @Override
    public String toString() {
        //DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

return
        "Symbol: " + symbol + " " +
        "Host: " + host.toString() + " \t " +
        "Time: " + time.toString() + " \t " +
        "Zone: " + zone.toString() + " \t " +
        "Method: " + method.toString() + " \t " +
        "Url: " + url.toString() + " \t " +
        "Response: " + response.toString() + " \t " +
        "Bytes: " + bytes.toString() + " \t " +
        "Referer: " + referer.toString() + " \t " +
        "User: " + user.toString();
    }
}

```

## Appendix D: Apache Log Input Adapter Configuration

The code is used to generate the log events and read through the EPL queries.

```
Package Logs;
import com.espertech.esper.client.*;
import java.util.StringTokenizer;
import java.text.*;
import java.io.File;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.BufferedReader;
import java.util.Scanner;

public class ApacheLogInputAdapter {

    public static class CEPLListener implements UpdateListener {
        public void update(EventBean[] newData, EventBean[] oldData) {
            System.out.println("EVENT! " + newData[0].getUnderlying());
        }
    }

    public static void main(String[] args) {

        // how the adapter is set up to retrieve log metadata from the log.java class

        Configuration cepConfig = new Configuration();
        cepConfig.addEventType("Apachecsv", Log.class.getName());

        // Configuration instance is then passed to EPServiceProviderManager to obtain a
        // configured Esper engine.

        //getProvider () method returns Instance of Esper Engine
        // code snippet shows how to send events to the engine

        EPServiceProvider cep = EPServiceProviderManager.getProvider("ApacheLogInputAdapter", cepConfig);
        EPRuntime cepRT = cep.getEPRuntime();

        EPAdministrator cepAdm = cep.getEPAdministrator();

        // the adapter loops over all EPL queries, this depends on the number of queries

        for(int queryCounter = 1; queryCounter <= 10; queryCounter++) {
            System.out.println("-----");
            System.out.println("PERFORMING QUERY " + queryCounter);
            System.out.println("-----");

            try {
                String query = new Scanner(new File("query " +
                    queryCounter + ".ep1")).useDelimiter("\\Z").next();
                cepAdm.destroyAllStatements();
                EPStatement cepStatement = cepAdm.createEPL(query);
                cepStatement.addListener(new CEPLListener());
            } catch (Exception e) {
                System.err.println("Error: " + e.getMessage());
            }
        }
    }
}
```

//The adapter reads through the Apache logs in the CSV form and generates EPL responses based on the EPL queries

```

try {
    File file = new File("Apacheaccesslogs.CSV");
    BufferedReader reader = new BufferedReader(new
FileReader(file));

    String line = null;
    StringTokenizer st = null;

    while((line = reader.readLine()) != null) {
        Log log = new Log("Apacheaccesslogs");

        st = new StringTokenizer(line, ",");

        while(st.hasMoreTokens()) {

            //DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
            try {

                //log.setTimestamp(dateFormat.parse(tokenizer.nextToken()));
            } catch(Exception e) {
                System.err.println("Error: " +
e.getMessage());
            }

            log.setHost(st.nextToken());
            log.setTime(st.nextToken());
            log.setZone(st.nextToken());
            log.setMethod(st.nextToken());
            log.setUrl(st.nextToken());
            log.setResponse(st.nextToken());
            log.setBytes(st.nextToken());
            log.setReferer(st.nextToken());
            log.setUser(st.nextToken());

            //System.out.println("LOG! " + log);
            cepRT.sendEvent(log);
        }

        //reader.close();
    }

    catch(Exception e) {
        System.err.println("Error: " + e.getMessage());
    }

}
}
}

```