

AN ARCHITECTURE FOR THE  
FORENSIC ANALYSIS OF WINDOWS SYSTEM GENERATED ARTEFACTS

NOOR HAYATI HASHIM

A submission presented in partial fulfillment of the requirements of the  
University of Glamorgan/Prifysgol Morgannwg for the degree of  
Doctor of Philosophy

November 2011



## ***Certificate of Research***

*This is to certify that, except where specific reference is made, the work described in this thesis is the result of the candidate's research. Neither this thesis, nor any part of it, has been presented, or is currently submitted, in candidature for any degree at any other University.*

*Signed*

.....  
*Noor Hayati Hashim (Candidate)*

*Date*

.....

*Signed*

.....  
*Professor Iain Sutherland (Director of Studies)*

*Date*

.....

## ABSTRACT

### AN ARCHITECTURE FOR THE FORENSIC ANALYSIS OF WINDOWS SYSTEM GENERATED ARTEFACTS

Computer forensic tools have been developed to enable forensic investigators to analyse software artefacts to help reconstruct possible scenarios for activity on a particular computer system. A number of these tools allow the examination and analysis of system generated artefacts such as the Windows registry. Examination and analysis of these artefacts is focussed on recovering the data extracting information relevant to a digital investigation. This information is currently underused in most digital investigations. With this in mind, this thesis considers system generated artefacts that contain information concerning the activities that occur on a Windows system and will often contain evidence relevant to a digital investigation. The objective of this research is to develop an architecture that simplifies and automates the collection of forensic evidence from system generated files where the data structures may be either known or in a structured but poorly understood (unknown) format. The hypothesis is that it should be feasible to develop an architecture that will be to integrate forensic data extracted from a range of system generated files and to implement a proof of concept prototype tool, capable of visualising the Event logs and Swap files.

This thesis presents an architecture to enable the forensic investigator to analyse and visualise a range of system generated artefacts for which the internal arrangement of data is either well structured and understood or those for which the internal arrangement of the data is unclear or less publicised (known and not known data structures). The architecture reveals methods

to access, view and analyse system generated artefacts. The architecture is intended to facilitate the extraction and analysis of operating system generated artefacts while being extensible, flexible and reusable. The architectural concepts are tested using a prototype implementation focussed the Windows Event Logs and the Swap Files. Event logs reveal evidence regarding logons, authentication, account and privilege use and can address questions relating to which user accounts were being used and which machines were accessed. Swap file contains fragments of data, remnants or entire documents, e-mail messages or results of internet browsing which reveal past user activities. Issues relating to understanding and visualising artefacts data structure are discussed and possible solutions are explored. The architecture is developed by examining the requirements and methods with respect to the needs of computer forensic investigations and forensic process models with the intention to develop a new multiplatform tool to visualise the content of Event logs and Swap files. This tool is aimed at displaying data contained in event logs and swap files in a graphical manner. This should enable the detection of information which may support the investigation.

Visualisation techniques can also aid the forensic investigators in identifying suspicious events and files, making such techniques more feasible for consideration in a wider range of cases and, in turn, improve standard procedures. The tool is developed to fill a gap between capabilities of certain other open source tools which visualise the Event logs and Swap files data in a text based format only.

## ACKNOWLEDGEMENTS

First, I am ever grateful to God, the Creator and the Guardian, and to whom I owe my very existence. Second, I would like to express my deep and sincere gratitude to my director of study, Prof. Iain Sutherland, for his support throughout the course of this thesis. I especially want to thank him for all the time he took to read and re-read this thesis so as to make it as good as possible. On the other hand, I also want to thank the people who served as evaluators in the evaluation phases of my thesis; research student at the Information Security Group, Faculty of Advanced Technology, University of Glamorgan.

On a different note, I am deeply indebted to my husband, Abdul Aziz Arshad, whose patient love enabled me to complete this work. Besides working to help with the financial needs, he finds the time to help with the household chores; cook, clean, and look after the children. He also provides suggestions and encouragement throughout the research and writing of this thesis. I also want to thank my daughter and son, Nur Fatihah Abdul Aziz and Muhammad Alfatih Abdul Aziz, who are the joy of my life, for letting Mak work on her research and thesis when she needed to do so.

Finally, a special thank you to my mum and dad, Allahyarhamah Saripah Ahmad and Allahyarham Hashim Abd Samad, to whom I dedicate my work. Although they are no longer in the world, their unfailing love and unwavering belief in my capabilities have contributed to my being able to successfully complete my doctoral study in University of Glamorgan. Last but certainly not least, my special gratitude is due to my sister, my brothers and their families for their loving support.

Noor Hayati Hashim

April 2011

# CONTENTS

<b>Table of Contents</b>	<b>vi</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xvi</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Background.....	1
1.2 Research Problem.....	6
1.2.1 Why This Research is Important .....	10
1.3 Research Hypothesis, Aim, Objectives and Questions.....	10
1.3.1 Hypothesis .....	10
1.3.2 Aim .....	11
1.3.3 Objectives .....	11
1.3.4 Information Extraction .....	12
1.3.5 Organising Data.....	13
1.3.6 Supporting Forensic Analysis.....	14
1.4 Scope of the Research.....	14
1.5 Research Methodology .....	15
1.6 Research Contributions.....	16
1.7 Organisation of the Thesis.....	17
1.8 Origins of Some of the Chapters.....	18
<b>2 LITERATURE REVIEW</b>	<b>19</b>
2.1 Forensics and Forensic Science.....	20

---

2.2	Computer Forensics and Computing .....	20
2.3	Computer Crime and Digital Investigation.....	22
2.4	Data Sources for Digital Evidence.....	26
2.4.1	Persistent Data .....	27
2.4.2	Volatile Data .....	30
2.4.3	Digital Evidence .....	31
2.5	Methodologies, Tools and Techniques .....	32
2.5.1	Digital Evidence Formats .....	33
2.5.2	Tools and Framework .....	38
2.6	An Overview of Computer Forensic Analysis and Data Collection .....	41
2.6.1	Logical Collection.....	42
2.6.2	Physical Collection.....	44
2.6.3	Analysis of Data .....	45
2.6.3.1	Timeframe or Temporal Analysis .....	47
2.6.3.2	Data Hiding Analysis.....	49
2.6.3.3	File Analysis.....	52
2.6.3.4	Relational or Link Analysis .....	56
2.7	Conclusion .....	56
<b>3</b>	<b>SYSTEM GENERATED ARTEFACTS AS FORENSIC OBJECTS</b>	<b>59</b>
3.1	Introduction .....	60
3.2	System Generated Artefacts .....	61
3.2.1	Event Logs As System Generated Artefacts.....	63
3.2.1.1	Event Logs Features .....	63
3.2.1.2	Evidentiary Value of Event Logs.....	68
3.2.1.3	Event Logs Tools and Related Issues .....	69
3.2.2	Swap Files As System Generated Artefacts .....	73
3.2.2.1	Swap Files Features.....	74
3.2.2.2	Evidentiary Value of Swap Files .....	75
3.2.2.3	Swap Files Tools and Related Issues.....	76
3.2.3	The Registry As A System Generated Artefact .....	77

---

3.2.3.1	The Registry Features .....	78
3.2.3.2	Evidentiary Value of the Registry .....	85
3.2.3.3	Registry Tools and Related Issues.....	87
3.2.4	Web Cookie Files As System Generated Artefacts.....	88
3.2.4.1	Web Cookie Files Features .....	88
3.2.4.2	Evidentiary Value of Web Cookie Files .....	90
3.2.4.3	Web Cookie Files Tools and Related Issues .....	90
3.2.5	Recycle Bin As A System Generated Artefact .....	91
3.2.5.1	Recycle Bin Features .....	91
3.2.5.2	Evidentiary Value of the Recycle Bin .....	93
3.2.5.3	Recycle Bin Tools and Related Issues.....	93
3.2.6	Internet Explorer Activity Files As System Generated Artefacts .....	94
3.2.6.1	Internet Explorer Activity Files Features.....	94
3.2.6.2	Evidentiary Value of Internet Explorer Activity Files.....	98
3.2.6.3	Internet Explorer Activity Files Tools and Related Issues.....	99
3.3	Digital Forensic Tools and Techniques .....	99
3.3.1	Commercial Forensic Tools.....	102
3.3.1.1	EnCase .....	102
3.3.1.2	Forensic ToolKit.....	104
3.3.2	Open Source Forensic Tools .....	105
3.3.2.1	Sleuth Kit and Autopsy.....	106
3.4	Problems Facing System Generated Artefacts Analysis .....	107
3.5	Conclusion .....	111
<b>4</b>	<b>SYSTEM GENERATED ARTEFACTS FORENSIC ANALYSIS SYSTEM DESIGN</b> .....	<b>113</b>
4.1	Introduction .....	113
4.2	Existing Structure and Requirements of Digital Forensic Investigations .....	114
4.3	Windows System Generated Artefacts Forensic Analysis System Architecture .....	115



---

4.4	System Analysis .....	126
4.5	Windows System Generated Artefacts Forensic Analysis (SAFTool) System Requirements .....	127
4.5.1	Functional Requirements.....	128
4.5.2	Security Requirements.....	130
4.5.3	Software Quality Requirements .....	130
4.5.4	Other Requirements.....	131
4.6	Windows System Generated Artefacts Forensic Analysis System Design .....	133
4.7	Operation and Control Specifications .....	138
4.8	Data Management.....	142
4.9	Object Oriented Approach to System Development .....	144
4.10	Conclusion .....	149
<b>5</b>	<b>SYSTEM GENERATED ARTEFACTS FORENSIC ANALYSIS SYSTEM IMPLEMENTATION</b>	<b>150</b>
5.1	Introduction .....	150
5.2	Software Development Tools.....	151
5.2.1	Integrated Development Environment.....	152
5.2.2	Object Oriented Programming.....	153
5.2.3	Database Management Systems (DBMS) and Connectivity .....	154
5.3	Implementation of the Architecture .....	156
5.3.1	Classes Identified.....	159
5.3.2	Interaction and State Behaviour .....	162
5.3.3	Data Management and the Database.....	165
5.3.4	SQL Statements .....	169
5.3.5	Fulfilling the Architecture's Other Requirements .....	171
5.4	Implementation Diagrams .....	175
5.5	Conclusion .....	178
<b>6</b>	<b>EVALUATION AND DISCUSSION</b>	<b>179</b>
6.1	Research Evaluation .....	181
6.1.1	Data Interpretation and Analysis.....	183
6.1.2	Experimental Setup .....	184

---

6.2	Comparison with Other Tools .....	192
6.3	Objective Evaluations Experiment of Forensic Analysis System .....	193
<b>6.4</b>	<b>Practical Experiments .....</b>	<b>194</b>
6.4.1	Experiment on the Event Logs with the Event Viewer .....	195
6.4.2	Experiment on the Event Logs with the evtstats.pl and lsevt.pl .....	197
6.4.3	Experiment on the Event Logs with the SAFTool.....	199
6.4.4	Swap Files Analysis Using WinHex .....	209
6.4.5	Swap Files Analysis Using the SAFTool .....	211
6.4.6	Analysis of the Results.....	218
6.4.7	A Comparison of the Results for the Event Logs and Swap Files Experiments .....	219
6.5	Subjective Evaluations Experiment of Forensic Analysis System .....	228
6.5.1	Subjects of Experiments.....	228
6.5.2	Structure of Experiments.....	230
6.5.3	Results .....	234
6.6	Additional Experiments .....	240
6.7	Conclusion .....	254
<b>7</b>	<b>CONCLUSIONS AND FUTURE WORK .....</b>	<b>256</b>
7.1	Summary .....	256
7.1.1	Information Extraction .....	258
7.1.2	Organising Data.....	259
7.1.3	Supporting Forensic Analysis.....	260
7.2	Conclusions.....	262
7.3	Issues and Future Work.....	264
	<b>Bibliography .....</b>	<b>266</b>
	<b>Appendices .....</b>	<b>296</b>
<b>A</b>	<b>Event Logs Data Structure for Windows Vista .....</b>	<b>296</b>
<b>B</b>	<b>Registry Data Structure .....</b>	<b>300</b>
<b>C</b>	<b>Use Cases .....</b>	<b>305</b>

---

<b>D</b>	<b>Sequence Diagrams</b>	<b>308</b>
<b>E</b>	<b>Full Class Diagrams</b>	<b>313</b>
<b>F</b>	<b>Full Database Tables</b>	<b>317</b>
<b>G</b>	<b>Full Dataset</b>	<b>323</b>
<b>H</b>	<b>Additional Dataset</b>	<b>326</b>
<b>I</b>	<b>Questionnaire</b>	<b>343</b>

# LIST OF FIGURES

2.1	Computer Investigation Model (Microsoft TechNet, 2007) .....	23
2.2	A Digital Evidence Bag Comprising Three Files (Turner, 2006) .....	35
2.3	An Overview of the Advanced Forensics Format (Garfinkel, 2006)....	36
2.4	An Overview of the Digital Evidence Exchange File Format (International Telecommunication Union, 2009).....	37
2.5	Layers of Analysis Based on the Design of Digital Data (in Carrier, 2005).....	41
2.6	An Overview of Correlation Between Types of Analysis .....	47
2.7	Methods for Analysing Data.....	49
3.1	Event Viewer Panes .....	70
3.2	An Example of the Opened Event Properties Dialog in Event Viewer	70
3.3	Windows Operating System History (Microsoft TechNet, 2007).....	73
3.4	Result of pagefile.sys Analysis .....	76
3.5	Registry Editor Showing Registry Values in the Value Pane .....	82
3.6	Hive Bin Structure (Morgan, 2009).....	83
3.7	The Last-logged-on User and Domain are Stored in the Key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\ CurrentVersion\Winlogon .....	86
3.8	Web Cookie File Content .....	90
4.1	High Level Description of Overall Architecture .....	120
4.2	Four Layers Architecture Applied to Overall Architecture.....	123
4.3	Four Layers Architecture Applied to Windows System Generated Artefacts Forensic Analysis System.....	123

---

4.4	Layer Architecture Relates to the Overall Architecture .....	124
4.5	Package Architecture Applied to the System.....	125
4.6	The System Analysis Model .....	126
4.7	Collaboration for the Windows System Generated Artefacts Forensic Analysis System .....	135
4.8	Objects for the Windows System Generated Artefacts Forensic Analysis System .....	138
4.9	Activity Diagram for Use Case Analyse New Artefact .....	140
4.10	Activity Diagram for Use Case Visualise Artefact Data .....	142
5.1	Package Diagram View for the Software Architecture.....	157
5.2	Structural and Data View of the Architecture.....	158
5.3	The Packages and Classes for the Windows System Generated Artefacts Forensic Analysis System.....	161
5.4	The Class Diagram for the Windows System Generated Artefacts Forensic Analysis System.....	161
5.5	Interface for the Use Case Analyse New Artefact .....	163
5.6	State Machine for the Visualise Artefact Data Window.....	164
5.7	Database Design for the Event Logs and Swap Files Structures .....	169
5.8	SQL to Create Tables for the Artefacts and Event Logs .....	170
5.9	Associations Between RelationalBroker Class and Classes from Other Packages .....	171
5.10	Plugin Framework.....	173
5.11	Component Diagram for the Windows System Generated Artefacts Forensic Analysis System.....	177
5.12	Deployment Diagram for the Windows System Generated Artefacts Forensic Analysis System.....	177
6.1	The Event Viewer Shows the Content of the Windows Event Logs..	196
6.2	The evtstats.pl Shows the Statistic from an Event Log File .....	198
6.3	The lsevt.pl Shows the Event Records from an Event Logs File.....	198
6.4	No Plugin Installed.....	201
6.5	The SAFTool Displays Active Menu When the Event Logs Plugin Has Been Installed.....	201
6.6	The SAFTool Displays Active Menu When the Swap Files Plugin Has Been Installed.....	202

6.7	The SAFTool Displays Active Menu After the Event Logs and Swap Files Plugin Have Been Installed.....	202
6.8	The SAFTool Displays the Event Log Files the User Can Choose to Analyse .....	203
6.9	The SAFTool Displays the Menu Option, Progress Window and Processing Time Message Window for the Event Logs Analysis .....	204
6.10	The SAFTool Displays the Menu Option and the List of Event Logs Files the User Can Choose to View Further .....	205
6.11	The SAFTool Displaying the AppEvent04.Evt Event Records.....	206
6.12	The Types of Events for the Event Logs .....	207
6.13	The Timeline of Event Logs Records.....	207
6.14	The SAFTool Allows the User to Generate A Report for the Event Logs Analysis. ....	208
6.15	The Report Generated for the Event Logs Analysis .....	208
6.16	The Hexadecimal Data Display, ASCII Text Display and Info Pane for WinHex. ....	210
6.17	The Find Text Function for WinHex.....	210
6.18	The SAFTool Displays the List of Swap Files the User Can Choose to Analyse .....	212
6.19	The SAFTool Displays the Menu Option, Progress Window and Processing Time Message Window for the Swap Files Analysis.....	213
6.20	The SAFTool Displays the Menu Option and List of Swap File Names the User Can Choose to Visualise the Swap Files .....	214
6.21	The Hexadecimal Data and ASCII Text Display Format for Displaying the Contents of Swap Files.....	215
6.22	The Square Block Diagram for the Density of Swap Files .....	216
6.23	The SAFTool Enables the User to Generate A Report for the Swap Files Analysis.....	217
6.24	The Report Generated for the Swap Files.....	217
6.25	Total Event Records Located for the evtstats.pl Script and SAFTool .....	220
6.26	Graph Showing the Number of Records Located With Processing Time .....	222
6.27	Comparison of Total Size of Data Shown by the WinHex and SAFTool .....	223
6.28	Total Size of Data Located Over Time for WinHex .....	225
6.29	Total Size of Data Located Over Time for the SAFTool .....	225

---

6.30	Participant's Satisfaction Level with the System Generated Artefacts Forensic Analysis Tool (SAFTool) .....	238
6.31	Total Event Records Located for the evtstats.pl Script and SAFTool in Consistency Test.....	242
6.32	Graph Showing the Number of Records Located With Processing Time in Consistency Test.....	244
6.33	Comparison of Total Size of Data Shown by the WinHex and SAFTool in Consistency Test.....	246
6.34	Total Size of Data Located Over Time for WinHex in Consistency Test .....	248
6.35	Total Size of Data Located Over Time for the SAFTool in Consistency Test .....	249
6.36	Participant's Satisfaction Level with Fifteen Participants in using the System Generated Artefacts Forensic Analysis Tool (SAFTool) .....	253

# LIST OF TABLES

2.1	Persistent Data Types (modified from Nolan et al., 2005) .....	28
2.2	Volatile Data Types (in Carvey, 2004; Bejtlich et al., 2005; and Nolan et al., 2005) .....	30
2.3	Data Hiding Methods .....	50
2.4	File Analysis Methods .....	53
3.1	System Generated Artefacts .....	62
3.2	Event Logs Organisation .....	64
3.3	Event Logs Header Structure .....	65
3.4	Event Log Records Data Structure .....	65
3.5	Values for Logon Types (Anson and Bunting, 2007) .....	68
3.6	Swap File Default Name and Location .....	75
3.7	Registry Files From Different Windows Platforms and Their Location .....	79
3.8	Registry Organisation (Anson and Bunting, 2007) .....	81
3.9	Registry Value Data Types (Anson and Bunting, 2007) .....	82
3.10	Registry Header Data Structures (Morgan, 2009) .....	84
3.11	Hive Bin Data Structures (Morgan, 2009) .....	84
3.12	Cell Data Structures (Morgan, 2009) .....	85
3.13	Web Cookie Files Location .....	89
3.14	Summary of Web Cookie Files Format (Jones, 2003) .....	89
3.15	Recycle Bin File Location .....	92
3.16	The Structure within the INFO2 File (Jones, 2003) .....	92



3.17	index.dat File Location .....	95
3.18	Fields in the index.dat File Header (Jones, 2003) .....	96
3.19	Fields in the HASH Table of index.dat File (Jones, 2003) .....	96
3.20	Relevant Fields in the REDR Activity Record (Jones, 2003).....	97
3.21	Fields in the URL and LEAK Activity Record (Haiping et al., 2009) .	98
3.22	Category of Tools by Britz (2004) and Svensson (2005) Studied.....	101
3.23	Comparison of Various Functionalities Provided by the Commercial and Open Source Tools Studied .....	109
4.1	Class Responsibility Collaboration for the Main Classes Contained in Windows System Generated Artefacts Forensic Analysis System....	136
5.1	Data Dictionary for the Structural and Data View Model Diagram .	159
5.2	Event Action Table for Figure 5.6.....	164
6.1	Requirements of the Architecture Revisited Illustrating the Architecture Satisfies Each.....	180
6.2	Numbering for Event Log Files and Swap Files.....	185
6.3	Event Logs Experimental Results for Experiment 2 .....	188
6.4	Swap Files Experimental Results for Experiment 2 Using WinHex	190
6.5	Swap Files Experimental Results for Experiment 2 Using Prototype Tool.....	190
6.6	Ability to Parse the Data Set and Display the Results .....	218
6.7	Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTTool .....	220
6.8	Processing Times for Event Viewer, lsevt.pl and SAFTTool .....	221
6.9	Total Size of Data for WinHex and the SAFTTool .....	223
6.10	Processing Time for WinHex and the SAFTTool .....	224
6.11	Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by WinHex .....	227
6.12	Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by the SAFTTool .....	227
6.13	Forensic Knowledge and Expertise of Human Subjects .....	230
6.14	File Size of File Subjects.....	230
6.15	Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTTool in Consistency Test .....	241

6.16	Processing Times for Event Viewer, lsevt.pl and SAFTool in Consistency Test.....	243
6.17	Total Size of Data for WinHex and the SAFTool in Consistency Test .....	245
6.18	Processing Time for WinHex and the SAFTool in Consistency Test.....	247
6.19	Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by WinHex in Consistency Test.....	250
6.20	Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by the SAFTool in Consistency Test.....	250
6.21	Forensic Knowledge and Expertise of Human Subjects of Fifteen Participants .....	251
6.22	File Size of File Subjects for Additional Experiments .....	252
A.1	Event Logs Header .....	297
A.2	Event Logs Chunk Header .....	298
A.3	Event Logs Record .....	299
B.1	Security Records (SK) Data Structures (Morgan, 2009).....	301
B.2	Key Records (NK) Data Structures (Morgan, 2009) .....	301
B.3	Subkey-lists Record Data Structures (Morgan, 2009) .....	302
B.4	Value Records (VK) Data Structures (Morgan, 2009) .....	302
B.5	Value-lists Records Data Structures (Morgan, 2009) .....	303
B.6	Normal Data Blocks Records Data Structures (Morgan, 2009) .....	303
B.7	Big Data Records Data Structures (Morgan, 2009) .....	304
B.8	Big Data Indirect Cells Records Data Structures (Morgan, 2009)....	304
G.1	Event Logs File Name and File Size of File Subjects .....	324
G.2	Swap Files File Name and File Size of File Subjects .....	324
G.3	Review of the Tasks Carried Out by Five Participants .....	325
H.1	Event Logs File Name and File Size of File Subjects in Consistency Test .....	327
H.2	Swap Files File Name and File Size of File Subjects in Consistency Test .....	327
H.3	Number of Records Retrieved for Event Viewer, lsevt.pl and SAFTool (First Run) .....	328
H.4	Number of Records Retrieved for Event Viewer, lsevt.pl and SAFTool (Second Run) .....	329

H.5	Number of Records Retrieved for Event Viewer, lsevt.pl and SAFTool (Third Run) .....	330
H.6	Processing Time for Event Viewer, lsevt.pl and SAFTool (First Run) .....	331
H.7	Processing Time for Event Viewer, lsevt.pl and SAFTool (Second Run) .....	332
H.8	Processing Time for Event Viewer, lsevt.pl and SAFTool (Third Run) .....	333
H.9	Total Size of Data for WinHex and SAFTool (First Run) .....	334
H.10	Total Size of Data for WinHex and SAFTool (Second Run) .....	334
H.11	Total Size of Data for WinHex and SAFTool (Third Run) .....	335
H.12	Processing Time for WinHex and SAFTool (First Run) .....	336
H.13	Processing Time for WinHex and SAFTool (Second Run) .....	336
H.14	Processing Time for WinHex and SAFTool (Third Run) .....	337
H.15	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the WinHex (First Run) .....	338
H.16	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the WinHex (Second Run) .....	338
H.17	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the WinHex (Third Run) .....	339
H.18	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the SAFTool (First Run) .....	340
H.19	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the SAFTool (Second Run) .....	340
H.20	Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the SAFTool (Third Run) .....	341
H.21	Review of the Tasks Carried Out by Fifteen Participants .....	342

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

An increasing crime rate in connection with the use of the Internet and computers has resulted in a growing demand for computer forensics, a field that is evolving to provide tools, techniques and systematic approaches to process and analyse digital evidence (Casey, 2004). Computer forensic investigators today analyse crimes ranging from computer security breaches to high impact crimes where damages can result with a large number of compromises, thereby causing monetary losses due to Internet threats and defamation (MyCERT, 2007).

The focus of this thesis is to examine the area of forensic tool development, to produce a prototype tool and to demonstrate the issues relating to tool creation. In comparison to other IT and computer professionals, it is common to see computer forensic investigators equipped with various tools (EnCase, FTK) in the same way as a network administrator is equipped with a range of software tools to diagnose faults and assess the security of a network. In order to obtain and process digital evidence, computer forensic investigators need to use tools, procedures and methods that are capable of

providing various functions in a forensically sound manner, and possibly, using layers of abstraction to detect features from large volume of data. In fact, the toolkits available to forensic investigator has been compared to a Swiss army knife by the National Institute of Standards and Technology (America) (NIST) (2006) where many tools provide very specific functionality that needs to be augmented by other tools during the course of a full investigation (Newman, 2007). Tools are usually designed to achieve a function or a particular range of functions to support the different phases of a digital forensic investigation. These key investigative phases are: system preservation, evidence searching, and event reconstruction (Carrier, 2006). Casey (2004) refers to these three phases of a digital investigation as: acquisition, analysis and presentation.

In the investigative phases, a tool is used during the acquisition phase, where the data is copied from the suspect storage device to either a trusted device or file. This tool must also preserve all of the data on the suspect's storage device, to prevent evidence from being altered or overwritten (ACPO, 2003). In the analysis phase, tools are used to examine the acquired data in order to identify pieces of evidence that support or refute a hypothesis regarding any incident. In the presentation phase, data from the analysis phase is arranged into a useful format; and a conclusion with related evidence from the investigation is presented. In all, computer forensic investigators need tools to identify, acquire, preserve and analyse data in a forensically sound manner. These requirements have inspired the production of various types of tools used during computer forensic investigations some of which are discussed in Chapter 2.

In investigative cases that involve computer systems, Carrier (2005) indicates that storage devices, specifically non-volatile devices such as hard disk drives, are important for analysing digital data. Based on studies of hard disk files (Gillam and Rogers, 2005; Alink et al., 2006; Garfinkel, 2006; Harms, 2006; Mee et al., 2006; Lee et al., 2007a; and Murphey, 2007), from the

physical to application level there are three levels of analysing data: volume analysis, file system analysis, and application analysis.

The analysis of the physical storage device usually begins with volume analysis. In volume analysis, data at volume level is examined to determine the location of file system, hidden data or other data. The contents of each volume, is usually a file system. File systems are a collection of data structures that are used by applications within the operating system to create, read, and write files. In most of the forensic work, file system analysis recovers the directory entries (Vlastos and Patel, 2007). Then, in file system analysis, the file system is analysed and resulting data fragments; metadata associated with files and file content. The structure of each file is dependant on the application or operating system that created the file. Application analysis can be described as the process of analysing the file contents in order to understand what is inside of a file. Analysis of the application level is important because this is the level where the various user activities and configurations relating to applications are recorded. Operating system files can be analysed to determine what programs were running and how a particular system may have been used (Carrier, 2005).

Considering the different possible application and operating system files a number of different forensic tools may be appropriate to analyse these files. Given the Windows registry as a forensic artefact, a computer forensic investigator may use different tools and procedures to analyse registry files. In the case of the Registry this is due to the fact that the registry file is a compound file that is comprised of multiple layers (Guidance Software, 2005).

In summary, the principle involved at the most basic physical media analysis sequence is: the disk is analysed to produce a stream of bytes. Followed by, the stream of bytes is analysed at the volume level to distinguish volumes. The volumes are analysed at the file system level to determine the files (including deleted material, but residing in the file system). The files are then analysed at the application level to determine user actions. Further

elaboration on existing approaches relating to file analysis is described in Chapter 2.

According to Carrier (2005), one of the basic challenges for any computer forensic investigator is the complexity of the data. There are numerous file types, each with a different structure and the structure is based on the application or operating system that created them. For example, a computer forensic investigator may require forensic evidence that illustrates the Internet usage in the Windows registry (Mee et al., 2006). This evidence may be uncovered using a specific methodology (often built into a tool), the tool presents them with the relevant hives that contain information in a tree-like structure.

In addition to physical media, investigators may also obtain evidence from other sources such as live memory analysis. Where the capture of live memory is not possible, some evidence of memory activity can be recovered from the Swap files left on the hard drive. The similarity between application analysis and memory analysis is that both requires information about internal data structure and fields layout (Schuster, 2006; Okolica and Peterson, 2010). In contrast to application analysis where a file is analysed to understand the contents, memory analysis examines the processes that were running in the memory. Both analysis deal with internal data structure and data organisation to work with. A file can be structured as one or more fields, containing data. These fields can be categorised into either information (the content of a file) or metadata (information on the structure or content of the file).

To interpret and analyse data, digital investigators must know how data is arranged in order to interpret the file content. Files are created by operating systems to facilitate quick access to applications and for a range of other purposes. The data structure of a file is a sequence of bytes, each sequence with a specific meaning and purpose. In system memory analysis, a process is the combination of some executable code, a virtual address space, and one or more threads of execution (data structure). Each process is

represented as an EProcess block. The EProcess block is a data structure that maintains various attributes of a process, as well as pointers to other attributes and data structures relating to the process (Ruff, 2007). The data structures (file content and EProcess block) need to be read and analysed by a computer forensic investigator when conducting an investigation.

A typical digital investigation of a computer system requires the individual analysis of numerous files. Reviewing files and interpreting their possible relevance to the case is time-consuming. This task becomes more complex as the volume of data increases, in particular when searching for specific files or content in large volumes of irrelevant files. The overwhelming volume of data found in modern systems which commonly includes Terabytes of data (Frauenheim, 2004), necessitates new efficient software-based tools, designed to deal with complexity, since data can take many interesting forms that require unique types of display and interaction. Fry (2007) suggests that the key goal of analysing data is to highlight its function in order of importance, and reveal the patterns and these functions that exist across multiple dimensions.

The proprietary nature and wide usage of the Windows system used (both desktops and servers) has made it a common source of evidence. Based on academic work into the investigative process in relation to the forensic analysis of Windows computer systems (Carvey, 2004; Bejtlich, 2005; Dongen, 2007; Luo, 2007; Murphey, 2007; Schuster, 2007), there are a number of possible artefacts for analysis, i.e. memory, application and file system. Morris (2003) suggests that system artefacts may be of evidential value when examining a Windows computer system. Several of these artefacts are created on Windows systems during normal operations without reference to the user and without the user's knowledge.

Since a number of artefacts register user activities and investigators are ignorant of this, there is a need to extract, analyse and present the artefacts in an intuitive and informative way. Accordingly, this thesis



addresses the analysis of Windows system generated artefacts. For the purpose of this thesis, a system generated artefact is defined as follows:

**Definition 1** *An artefact is a single file which may potentially contain evidence that will have been created as a routine function of the various types of computer operating systems. For example, an artefact which has been created as a repository for deleted files, contains information stored in records so that the original information about the file may be restored, such as the file name.*

In today's world, where the number of crimes committed using computers continues to increase, a need exists for advanced forensic software tools which allow computer forensic investigators to follow digital tracks left by persons committing illegal activities. According to Volonino et al. (2007), "*plain text documents, log files, or even system files may contain traces of this evidence*". Using visualisation techniques in displaying information about computer data helps forensic specialists in searching suspicious files (Teerlink and Erbacher, 2006; Vlastos and Patel, 2007; Read et al., 2009). The work described in this thesis concerns the architecture that supports the extraction, analysis and presentation of the system artefacts. The Windows operating system has been considered for the work due to its popularity, and to ensure evaluation of the work can be performed adequately by being able to gain access to a sufficiently large set of test data (see Chapter 7).

## 1.2 Research Problem

Recent technological advancements in computer hardware have had a significant impact on the size and cost of hard disks. It is known that hard disk capacity is increasing and the price of data storage is decreasing (Hitachi, 2008). Digital investigators can be overwhelmed by the vast number of files contained on a single modern hard drive where capacities at the time of writing are around 2 terabytes and are estimated to reach 4 terabytes in size by 2011 (Brown et al., 2005). In the case where only a portion of the digital evidence on the computer is of interest (e.g. a log file), it is more practical to

search the computer immediately and just take the information required (Craig et al., 2005). Extracting only essential files is easier, faster, and less expensive than copying the entire contents and can also solve the potential business process problem of shutting down a large server (Sommer, 1998). Nevertheless, in extracting limited sets of files there is a risk that digital evidence will be overlooked or damaged during the collection and preservation process. Given the risks of collecting only a few files, it is necessary to impose a strict approach that will maintain the integrity of the digital evidence acquired.

Acquiring a certain file directly from a live system may be impossible when an operating system kernel prevents file access (such as a swap file). However, it is still possible to access this file using a specially crafted driver, or a special device, for example, Filter\_1 by New Technologies, Inc. ([www.forensic-intl.com](http://www.forensic-intl.com)) (Schweitzer, 2003). There are also a number of commercial tools that are able to copy the swap file of a running system. For example, the software utilities like Norton Commander or Norton DiskEdit. However, as with a number of these system artefacts the easiest way to collect the swap file is to unplug the system and export the file from a drive.

System generated artefacts represent valuable sources of evidence and are increasingly the focus of investigation and legal discovery as they are generated by the system and are not readily visible to the common user, which also makes it more plausible that they have not been altered (Volonino et al., 2007). Examples of data that can be recovered and examined are Internet activity and temporary backup files, passwords and deleted files. Hence, ignorance on the part of the digital forensic investigator concerning the location, format, existence and content of these system artefacts can result in important evidence being lost. Therefore, system generated artefacts as in Definition 1 make analysis of system generated artefacts to be easily realised and understood by digital forensic investigator is one of the most exciting areas to be explored in order to improve the effectiveness of digital investigations.

Analysing a system generated artefact is usually achieved by translating a stream of bytes into a usable data structure. An important question related to data structure is how to access and manipulate the data structures and fields that composed a data structure. On top of the technical challenges of locating and interpreting information, digital forensic investigators face the challenge of interpretation.

During the analysis phase, correct interpretation requires not only data structure with their fields' layout, but also data organisation. However, digital forensic investigators, either experienced or inexperienced, should ideally be free from the issues of dealing with the concepts of data structure and data organisation. A data structure and a data organisation are defined below:

**Definition 2** *A data structure is an organisation of data including structural relationships* (Adamson, 1996).

**Definition 3** *“A data organisation is a way to represent information in a data structure, together with algorithms that access and/or modify the structure”* (Knuth, 1997).

Digital forensic investigators have to sift through hundreds of thousands of files and analyse each file on a computer system. Normally, a typical file needs to be analysed, either by the application that created the file or by a suitable specialised viewer. For example, a JPEG file requires an image viewer to correctly interpret the JPEG file format. As a result, the examination process of one file can significantly differ from the examination process of another file. For these reasons, as previously indicated, the forensic investigator needs a large toolkit to process and interpret evidence. An investigator may also benefit from further additional tools, ones that are capable of visualising the evidence extracted from a case, given the large volumes of data which need to be sifted through during an investigation. According to Ayers (2009) (as cited in Roussev and Richard, 2004), *“existing tools are failing to keep pace with the increasing complexity and evidential volumes of modern computer forensic investigations”*. These problems have been addressed in other fields: such as network security, by using visualisation

methods (Read et al., 2009); for reverse engineering of binary and data files, by using visualisation methods and text based methods (Conti et al., 2008); and in overcome the limitations of existing digital evidence presentation methods, and text or command line utilities, by using of 3D visualisation techniques (Vlastos and Patel, 2007). Due to the size and complexity of the data sets associated with network traffic, particular patterns indicating evidence of a network intrusion may often only be detected when the data is displayed visually to a human operator. According to Teerlink and Erbacher (2006), *“In particular when examining the advanced techniques developed during the last few years for hiding, wiping, encrypting and deleting digital data. It has also been suggested that visually displaying information related to a digital investigation can reduce the time required to identify evidential material”*.

This thesis suggests an approach for extracting, analysing and representing digital evidence from system generated artefacts. This includes artefacts for which the internal arrangement of data is either well structured and understood or those for which the internal arrangement of the data is unclear or less publicised; for the purpose of this thesis these are referred to as ‘known’ and ‘not known’ data structures. Ideally it should be possible to extend the proposed architecture in the future to extract and analyse other types of artefact if an extensible architecture was used, and possibly, a design that would only require only the relevant parser to interpret the different data structure.

Such system generated artefacts forensic analysis tool could provide further evidence as to what may have occurred during an investigation, for instance, when checking the Windows system for signs of compromise. Such a tool could also provide a passive forensic analysis system solution that could visualises the system’s activity for signs of compromise and analyse and visualise data from the registry, event logs, recycle bin, Internet explorer activity file, swap files and more.

### 1.2.1 Why This Research is Important

This research is important in that it will support digital forensic investigators in answering questions that arise prior to analysing a system generated artefact. Such questions include: *what is the internal data structure (digital evidence) of the artefact – what can be revealed when we decode the artefact?, and which method is most appropriate for visualising the extracted data? For example, is the artefact content suitable for displaying in text or command line utilities?*

In addition, a digital forensic investigator may decide to examine or display the data in different ways, an alternative method of presentation may be preferable to a specific investigator or when considering different types of investigation. For instance, in some investigations, the content of an image file may be the significant element of a case, in other investigations it may be the metadata associated with the image file. Therefore, a digital forensic investigator whose intention is to use the revealed artefact's contents as evidence in developing their report may like to know if the artefact's contents can be highlighted in order of importance, or if there are relationships with other files.

## 1.3 Research Hypothesis, Aim, Objectives and Questions

### 1.3.1 Hypothesis

This thesis proposes that it should be feasible to develop an architecture that is able to integrate forensic data from known and not known internal data structures of system generated artefacts by the Windows operating system and to design and implement a proof of concept prototype tool, with appropriate example artefacts.

### 1.3.2 Aim

The key aim of this project is to develop an architecture that simplifies and automates the collection and extraction of forensic evidence for known data structure and not known internal data structure system generated artefacts by the Windows operating system. The project plans to develop a prototype tool that is capable of visualising known data structures of appropriate artefacts in such a way that the investigator can easily see what data is available within these areas of the Windows. This architecture should aid investigators in the forensic analysis of Windows system generated artefacts.

### 1.3.3 Objectives

The objectives of the project are summarised below:

- I. Research
  - a. A review of system generated artefacts and the use of this type of artefact in computer forensic analysis.
  - b. Current state-of-the-art tools regarding accessing and visualisation of system generated artefacts.
- II. Implementation
  - a. To develop an architecture to deal with system generated artefacts, and to integrate data from different sources.
  - b. To implement a prototype tool, based on the architecture, capable of visualising a selected subset of system generated artefacts files in such a way that the investigator can easily access and extract available data.
- III. Evaluation

The evaluation process would involve:

  - a. Comparison of the proposed system would be made against current state-of-the-art tools used.

- b. An evaluation using appropriate test data sourced from blind sourced second-hand hard drives.

The research hypothesis will be verified by developing a prototype digital forensic analysis tool that is built upon open-source applications and that presents users with an interface that displays the content of a specific file. The research is structured around three central questions discussed in Sections 1.3.4, 1.3.5 and 1.3.6.

Performance of the digital forensic analysis software/system/tool is evaluated through (1) objective analysis and (2) subjective experiments. The objective analysis involves quantifying the accuracy of the data extracted from the forensic analysis software/system/tool upon receiving an artefact and displaying the content of that artefact to support and improve the artefact forensic analysis system. In one experiment, the effectiveness of the visualisation function of the proposed digital forensic analysis software/system/tool is analysed through experiments. These used a series of scenarios constructed from a selection of Event logs and Swap files information obtained from Glamorgan University's Computer Forensic laboratory in which forensic images were taken during a disk study (Jones et al., 2009). Another experiment considers verifying functionalities and usefulness of the software application is done through eliciting expert opinion. The objective is to verify the software's capability of visualising known data structure of Event logs and not known data structure of Swap files in such a way that the investigator can easily see what data is available within these areas of the Windows operating system.

#### 1.3.4 Information Extraction

**Question 1:** *How can we extract information from an artefact?*

One of the first challenges that an artefact forensic analysis system has to cope with is how the data have been stored in a specific file / file system. This is

achieved by interpreting the data structure as this describes the relevant data and metadata portions of the file and file system. When we want to read data from the storage device, we determine where the data starts and then refer to its data structure to determine what the information of value is stored. The challenge is to interpret this data structure: each field has a size and name, although the size and name information is not saved with the data structure, and represents the information in a data structure. An additional challenge is to find abstraction data structures that are not explicitly available in a file and can be used to represent a file. Thus, appropriate parsers are required to extract different types of data structure that are explicitly and implicitly contained in a file.

### 1.3.5 Organising Data

**Question 2:** *How can we organise and integrate the various data structures to improve the correlation of the data obtained?*

Upon identifying data structures (known and not known), how can these data structures be integrated? How can data be used for data retrieval, further analysis and data representation? An issue that should be considered is the flexibility of the proposed mechanism. The mechanism should be flexible enough so that additional data structures can easily be incorporated into a common data store and this should include the facility to search through the data. Keyword searching is used to search for texts or hexadecimal values that the user may be have interest. At the same time, mechanisms to ensure that no data is altered should also be considered, to ensure forensic integrity is maintained. This is usually achieved in forensic tools by the use of a hash function which is used to verify the integrity of the original media before and after imaging and used to verify the integrity of working copies of the original media (Casey, 2000).



### 1.3.6 Supporting Forensic Analysis

**Question 3:** *How can we use the information obtained in the first two questions to support and improve forensic analysis?*

Several issues have to be addressed before the information identified in Question 1 can be used to improve forensic analysis: How to deal with data presentation that represents the different context of understanding data. Also, when the user gains useful insight about the file, how can information be inferred from fields extracted from a data structure? In order to identify the benefits of incorporating visualisation as a function into the tool, we need to perform relevant analysis on the techniques that can be used to expose particular aspects of the data.

## 1.4 Scope of the Research

This research works on artefacts from the Windows family of operating systems. An artefact is a single file or an area containing evidence that has been created as a routine function of the various types of computer operating systems, and is a computer-created file. It is categorised into two classes of file format that is a known and not known file format. A not known file format artefact is a “flat” file and will not have any fields. These types of artefact usually have operations that may only be performed on objects of this type. Users are allowed to examine and manipulate such objects using only these operations, not knowing how the objects have been implemented (Schneider and Bruell, 1998).

This research limits its work to the Windows XP and Vista operating systems as these two versions of the operating system were the most common systems in use at the beginning of this work. They were also the two most commonly encountered in the disk study analysis conducted at the University of Glamorgan (Jones et al., 2009), thus a significant body of test data will be

available. In terms of operating system artefacts, for reasons outlined in Chapter 3, this research focused on event logs file and swap files. These artefacts are selected based on their internal data structure and the characteristic of the file. It should however be noted that it is intended that the system is intended to be able to readily adapt to new files. The proposed system focuses on examination of a file by extracting, analysing, and displaying file information visually. It does not focus on the preservation of hard disk image, case management facilities or other processes involved in a computer forensic investigation.

## 1.5 Research Methodology

The following sections of this thesis describe the research approach used to address the research hypothesis outlined in this chapter (Chapter 1). It explores possible research design methodologies and then assesses the possible evaluation methods that could be applied both to the proposed architectural design and the prototype software. In terms of research methodology and research design Lakatos (1978) and Kuhn (1996) define research as “*an activity that contributes to the understanding of a phenomenon*”. The phenomenon is a set of behaviours of an entity that is of interest to a researcher or a particular research community. Understanding a phenomenon provides knowledge that enables prediction of the behaviour of some aspect of the phenomenon. Research methods or techniques are the set of activities a research community considers appropriate for the production of knowledge. Research can be a key tool in informed decision-making and can be central to determining what should be done, what can be done, how it will be done, and how well it has been done (O’Leary, 2005).

According to Leedy and Ormrod (2005), “*a research design is a general strategy for solving a research problem*”. According to Kumar (2005) as cited in Kerlinger (1986), “*A research design is a plan, structure and strategy of investigation so conceived as to obtain answers to research questions or*

*problems. The plan is the complete scheme or programme of the research. It includes an outline of what the investigator will do from writing the hypotheses and their operational implications to the final analysis of data.*" In planning and designing a research study, the researcher addresses the research problem through certain methodologies that are particularly appropriate to the nature and type of data the investigation of the problem requires (Kumar, 2005). In addition, data and methodology interdependency is where the methodology to be used for a research problem take into account the data that needs to be collected to address the research aims, objectives, and questions (Leedy and Ormrod, 2005).

In the application development phase, the design is converted to a functional code. Development of a program (implementation) is the process where program functions and activities are put in place (BJA, 2010). The implementation encompasses coding, unit testing, and test-case definition activities. The proposed approach is implemented as a working tool to support system design practice. It is implemented as a tool to demonstrate the solution for the hypothesis of the research, which is a new architecture able to analyse and visualise information extracted from Windows system generated artefacts.

## 1.6 Research Contributions

Several of the results set this research apart from other related studies. The research proposes an overall solution and validates a new architecture that supports digital evidence examination and is applicable to a wide range of internal data structures found in Windows system artefacts. The experiment shows that this approach provides improved support for a number of artefact forensic analyses. The contributions of this thesis are as follows:

- An architecture for extracting artefacts contained in hard disk images. The artefact data is parsed separately and later combined into a single representation. Such an approach can be extended to include other data

from various areas within the Windows system as identified and/or required by the user.

- An architecture that is capable of visualising data contained in an artefact in such a way that the investigator can easily see what data is available within these areas of Windows operating system.
- The provision of an open source architecture. Further, this architecture will be as extensible and flexible as possible: for future improvements, research and addition of features; and integration of the implemented system into a wider forensic tool.
- Visualisation is the method chosen to understand and communicate information. A prototype tool has been implemented for visualising the Event logs and Swap files data. Visualisation techniques used is means to reveal patterns and show features in order of their importance.

## 1.7 Organisation of the Thesis

The thesis has seven chapters, including this introductory chapter which covers the background to this research, followed by the research problem, research hypothesis and questions, scope of the research and research contributions.

**Chapter 2** covers the literature review on the theoretical foundation of computer forensics, the processes involved in a digital forensic investigation that involves digital evidence, and the spectrum of physical storage media analysis. It also investigates the theory and practice of examining artefacts extracted from a Windows operating system.

**Chapter 3** focuses on information contained in system generated artefacts. This includes the data structure and data organisation of system generated artefacts. It also explains the possible forensic value and importance of these

system generated artefacts. The state-of-the-art tools available to the digital investigator are also examined and discussed.

**Chapter 4** is central to this work and describes the design process of the proposed architecture. It introduces the proposed architecture which deals with the complex structure of Windows system generated artefacts. The architecture requirements and objectives are defined based on the priority of the requirements identified in Chapters 2 and 3, and from the highlighted issues of the current state-of-the-art tools.

**Chapter 5** provides details of how artefacts with known and not known data structures (previously identified in Chapter 3) are incorporated into the visual system generated artefacts forensic analysis system (SAFTool). This chapter explains the implementation of the architecture and the development of the prototype, and discusses various techniques used to implement the architecture.

**Chapter 6** details the results and describes the evaluation of the visual system generated artefact analysis system, which includes a qualitative review of the input from experts in the field of forensics. In addition, lessons learned from the evaluation are also presented and discussed in this chapter.

**Chapter 7** concludes the thesis by summarising the contributions made and discussing future research directions.

## 1.8 Origins of Some of the Chapters

Parts of this thesis have been published previously. Portions of Chapter 3 are based on the work presented in Hashim and Sutherland (2007). Portions of Chapter 4 are taken from Hashim and Sutherland (2010).

## CHAPTER 2

# LITERATURE REVIEW

This chapter reviews the background for the development of the architecture and prototype forensic analysis tool. The development of an architecture and prototype forensics analysis tool involves understanding the current best practice and literature available on the theoretical foundation of computer forensics and the processes involved in a digital forensic investigation that involves digital evidence, and the full spectrum of physical storage media analysis. It also investigates the theory and practice of examining artefacts extracted from the Windows operating system. The chapter starts by presenting concepts from forensic science and computer science that can be used to collect, examine and analyse digital evidence stored on a computer. Although this chapter focuses on the background of computer forensics analysis, it also discusses some issues related to the analysis of digital evidence and the representation of digital evidence since they form the basis for the research described in subsequent chapters. The specific focus of this chapter is to review the key issues associated with the forensic analysis of a computer system.

## 2.1 Forensics and Forensic Science

Casey (2004) defines forensics as “*a characteristic of evidence that satisfies its suitability for admission as fact and its ability to persuade based upon proof*”. Forensic techniques can be used to recover and analyse latent evidence such as fingerprints left on doors, DNA recovered from blood, or, in computer or digital forensics, the files on a hard drive. Casey (2004) also indicates that forensic science provides tools, techniques and a systematic approach to process and analyse digital evidence and use this evidence to reconstruct what occurred during the perpetration of a crime, with the ultimate purpose of linking an offender, victim and crime scene. The use of forensic science provides a body of proven scientific investigative knowledge, techniques and methods for formulating and testing hypotheses concerning what may have occurred during unauthorised or criminal activity.

## 2.2 Computer Forensics and Computing

A number of core concepts from forensic science can be transferred into computer forensics, for example, offering carefully tested methods for processing and analysing digital evidence. According to Guillermo et al. (2007), “*computer forensics combines elements of law and computer science to collect and analyse data from computer related systems in a way that is admissible as evidence in a court of law*”. The use of a computer to create and store information leaves behind ‘electronic fingerprints’ that can be fundamental to determining the outcome of a criminal case (Anderson, 2005).

In the literature, several definitions of computer forensics discussing how a computer may be related to a digital investigation have been found (Caloyannides, 2001; Kruse II and Heiser, 2002; Vacca, 2002; Carrier, 2003; Mohay et al., 2003; Britz, 2004; Solomon et al., 2005). According to Mohay et al. (2003), “*computer forensics relates to the investigation of situations where there is computer-based (digital) or electronic evidence of a crime or suspicious*

*behaviour, and the crime or behaviour may be of any type, quite possibly not otherwise involving computers*". Computer forensics therefore involves the preservation, analysis and interpretation of computer data and more specifically, it establishes procedures for recovery, preservation, and analysis of digital evidence (Britz, 2004). According to Mohay et al. (2003), "*it is also concerned with the analysis of any information stored by, transmitted by or derived from a computer system in order to determine the validity of hypotheses which attempt to explain the circumstances or cause of an activity under investigation*". According to Solomon et al. (2005), "*In terms of the investigative process forensic computing is the process of identifying, preserving, analysing, and presenting digital evidence in a manner that is acceptable in a legal proceeding*". According to Britz (2004), the importance of computer forensics processes protecting digital evidence from alterations, damage, and data corruption by providing mechanisms for evidence duplication to enable creation of forensically sound images that are useful for data analysis.

According to Vacca (2002), "*Computer forensics is about evidence from computers that is sufficiently reliable to stand up in court and be convincing*". The process begins with the acquisition of digital evidence, when information is collected or stored in anticipation of being examined and computer forensics ensures the preservation and authentication of computer data which can be easily altered, erased or subjected to claims of tampering if it is not properly handled. According to Schweitzer (2003), "*An important aspect of computer forensic analysis is the recovery and analysis of deleted files and other forms of compelling information that are normally invisible to the user*".

Caloyannides (2001) reviews computer forensics as "*the collection of techniques and tools used to find evidence in a computer*". Kruse and Heiser (2002) emphasise the responsibility of computer forensic specialists as being "*to follow clear, well-defined methodologies and procedures, and flexibility encouraged when encountering the unusual*". According to Carrier (2003), "*the*



*term digital forensics is commonly used and has historically been used to describe a much more involved process where the investigator must trace user activity and cannot provide a simple yes or no answer”.*

Computer forensics can be regarded as a reactive field because the forensic collection and analysis of data is shaped by the development and use of new technology and applications.

## **2.3 Computer Crime and Digital Investigation**

Volonino et al. (2007) employs the term computer crime to describe both information crime and high-tech crime, terms that are used interchangeably by most people, the courts, and the legal system. These terms can be used to refer to the two categories of offenses that involve computers: Firstly where the computer is a target, so the computer or its data is the target of a crime. In the second case the computer as an instrument; where a computer is used to commit the crime.

According to Britz (2004), *“the first major publically noted computer crime occurred in 1986, when an accounting error of less than one dollar was investigated by employee at the University of California at Berkeley”*. In this case although it appeared initially as an insignificant accounting error this actually related to a vulnerability that existed within the data system that was exploited by a hacker who was able to move about the system with remarkable ease and relative impunity. Child pornography, web defacement, fraud investigations, corporate investigations and hacking are examples of computer crime where computers are involved at some point in the case. However, there are three, not mutually exclusive, categories of computer crime: targets, means, and incidentals. According to Britz (2004), *“In fact, a computer itself a piece of evidence, can be processed to identify thousands of pieces of digital evidence and each piece of digital evidence can be analysed to identify ownership, location and timing”*.

Instead, the term digital investigation is apparently used to describe “a process that uses science and technology to analyse digital objects and that develops and tests theories, which can be entered into a court of law, to answer questions about events that occurred” (Carrier, 2005).

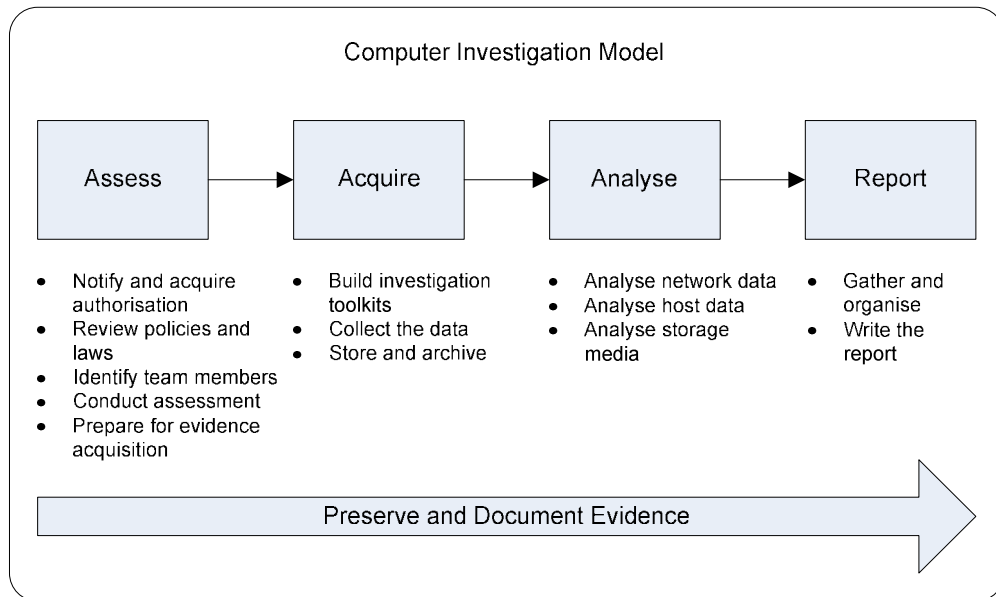


Figure 2.1: Computer Investigation Model (Microsoft TechNet, 2007)

There are a number of possible model that outline the process of an investigation including (Carrier and Spafford, 2003; Mandia et al., 2003; Mohay et al., 2003; DoJ, 2007; Microsoft TechNet, 2007). However all of these models have four keys phases in common: Assess, Acquire, Analyse, and Report (Figure 2.1): when an investigator is faced with a possible digital crime, the investigator should first analyse the scope of investigation and formulate the action to be taken. An example outcome of this assessment phase is a detailed document containing all of the information that is considered relevant about the situation and provides details about how systems might be affected and a proposed course of action.

Once a course of action has been determined from the Assessment, then as shown in Figure 2.1 the Acquire Phase is implemented to gather, protect and preserve the original evidence. This process of acquisition needs an appropriate collection of hardware and software tools to acquire data during the investigation. Such a toolkit may contain a laptop computer with suitable software tools and also backup media which can be write protected. This toolkit should be created in advance from a collection of tried and tested tools so that the investigator is familiar with the tools features and more importantly the tool limitations before they are used in an actual investigation. Digital evidence collection can be performed on either 'live' or 'dead' systems. In live systems investigations, where the computer is powered on during part of the investigative phase, the main focus is usually evidence relating to volatile memory or continuing network activity. The investigation may also involve the collection of specific data from very large systems. In some cases this may rely on some areas of the host operating system of the computer being investigated to support the analysis. However, in 'dead' systems investigation, where the computer is powered off and the storage media (usually the hard drive) is copied and analysed, trusted applications in a trusted operating system are used to find evidence. The use of an identical copy and trusted software tool is to support the admissibility of the evidence in a court of law. When important data are saved during an investigation, a MD5 or SHA1 hash value of the content of a file is computed and to show that the copied data is not altered and to provide future reassurance of the file's authenticity. Evidence is stored and archived in a way that ensures its safety and integrity. There are three options when collecting digital evidence from a computer: just copying the information needed by creating a sparse copy of a folder or file according to Collins (2008), sparse means not dense, copying everything by creating a bit-stream disk-to-file and copying

everything by making a bit-stream disk-to-disk copy (Nelson et al., 2004).

Once evidence has been collected the data requires analysis to provide an interpretation and possible meaning for any network, host or removable media data that has been collected as part of the process (Figure 2.1). Different approaches can be applied during the analysis in line with recommended best practice. Different forms of analysis include timeframe analysis for the timeline activities on victim's computer that showing email correspondences, online chat session (Casey, 2004), data hiding analysis for the deleted files and email that have been purposely hidden (Nelson et al., 2004) and file analysis for searching credit card numbers that could be stored in a spreadsheet or database data (Solomon et al., 2005). Furthermore, the specific type of analysis performed depends on the goal of the investigation and to some degree on what evidence the investigator expects to find based on the charges brought against the suspect.

In the final section of the model outlined in Figure 2.1 the Report Phase, the gathered information and documentation collected throughout the investigation is interpreted to formulate conclusions based on the evidence found.

The course of action recommended by most guidelines (Middleton, 2002; Shinder and Tittel, 2002; Mandia et al., 2003; Schweitzer, 2003) suggests that irrespective of whether the case will end up in a court of law or form part of internal disciplinary proceeding, the goals of a forensic investigations are to: conduct structured investigation, preserve and secure electronic data using methods that can withstand the court of law, obtain all relevant data, minimises the cost and business disruption to the host organisation, and to integrate any computer related digital evidence into appropriate legal proceedings (Cummings and Lowry, 2003).

## 2.4 Data Sources for Digital Evidence

Data is the basic form of information that is collected, analysed and interpreted to create knowledge in computer forensics. Electronic data is easily created, hidden and manipulated, and deleted. These electronic data can be found through the forensic examination of the computer storage media, typically the hard disk drive. A hard disk can contain a myriad of different files systems and file types. Digital forensics, in many cases can provide a substantial body of evidence: state of mind and knowledge of, access to specific information and about a computer user's activities (Howell, 2005).

There are a number of ways of describing the types of data that are collected in a typical forensic investigation (Guillermo et al., 2007). Volatile data can be described as data that is stored in memory or that will be lost when the computer is turned off. The term persistent or non-volatile data is used to mean data stored on a logical hard drive and is preserved when the computer is powered off. Persistent data includes active data, temporary data and ambient data (or residual data) and archival data. According to Hailey (2003) active data is the information that can be accessed through the file system. This refers to data files, programs, and files used by the operating system. Archival data is data that has been backed up and stored. This could consist of backup tapes, CD-ROMs, USB Storage devices or entire hard drives. Mohay et al. (2003) states "*ambient data is typically used to describe data that is stored in non-traditional computer storage areas, not accessible at the logical or application level, created by operating system and applications in the background during operation, and typically includes deleted files, file slack, volume slack, Windows swap file, unallocated space, stored printer images and Internet artefacts*". Lewis (2004) and Newman (2007) also support this statement.

### **2.4.1 Persistent Data**

Persistent data is important because of the high volume of material available and subsequently most forensics evidence is found in persistent data. Examples include documents, emails, web activity and deleted files. Persistent data for Windows and Unix/Linux can be compiled as in Table 2.1. Persistent data includes operating system generated artefacts such as logs, files, lists, passwords, caches, history and recently used lists. Some is in plain text, some is obscured and some is encrypted. These system generated artefacts are the result of the evolving and increasing complexity of the different forms of computer operating systems. To simplify the user interface of many operating systems, and to deal with the issue of multiple users, typically the modern operating system has to store increasing amount of information relating to a user, their actions, preferences and credentials. Such types of data or information can be referred to as operating system generated artefacts. The most important thing about operating system generated artefacts is that they can be used to as evidence to identify users and their computing activities (Guidance Software, 2005).

Table 2.1: Persistent Data Types (modified from Nolan et al., 2005)

	Data Avenue By Operating System	
	Windows Examples	Unix/Linux Examples
System Files	Basic input/output system (BIOS) setup Boot Records <ul style="list-style-type: none"> <li>- Master boot record</li> <li>- Volume boot sector</li> </ul> Event/System Logs <ul style="list-style-type: none"> <li>- MAC times (mtime – time of last modification, atime – time of last access, ctime – time of last status change)</li> </ul> System Registry	Basic input/output system (BIOS) setup Boot Records <ul style="list-style-type: none"> <li>- Master boot record</li> <li>- Volume boot sector</li> </ul> Event/System Logs <ul style="list-style-type: none"> <li>- /etc/syslog.conf</li> </ul> Timestamps <ul style="list-style-type: none"> <li>- Modification time</li> <li>- Access time</li> <li>- Change of status</li> <li>- Deleted time</li> </ul>
Temporary Files	*.tmp Spool File <ul style="list-style-type: none"> <li>- *.shd</li> <li>- *.spl</li> </ul>	- Spool File <ul style="list-style-type: none"> <li>- /var/spool/lpd</li> </ul>
Web Artefacts	Internet Explorer <ul style="list-style-type: none"> <li>- Bookmarks (favourites) C:\Documents and Settings\[user]\Favourites</li> <li>- Cookies C:\Documents and Settings\[user]\Cookies</li> <li>- URL history C:\Documents and Settings\[user]\Local Settings\History</li> </ul>	Each Linux flavour, build, or browser might put the data in a slightly different place.

Table 2.1: Persistent Data Types (modified from Nolan et al., 2005)(continued)

	Data Avenue By Operating System	
	Windows Examples	Unix/Linux Examples
Web Artefacts	<ul style="list-style-type: none"> <li>- Temporary Internet files</li> <li>- C:\Documents and Settings\[user]\Local Settings\Temporary Internet Files</li> </ul> <p>Registry Hive</p> <p>HKEY_CURRENT_USER\Software\Microsoft\Internet\Explorer\Typed URLs</p> <p>Netscape</p> <ul style="list-style-type: none"> <li>- Cookies</li> <li>- URL history</li> <li>- history.dat (a Berkeley DB file)</li> <li>- Temporary Internet files</li> <li>- Web Cache</li> </ul> <p>index.db (a Berkeley DB file)</p>	
File Recovery	<p>Deleted data</p> <ul style="list-style-type: none"> <li>- Files</li> <li>- Emails</li> </ul> <p>Slack space</p> <ul style="list-style-type: none"> <li>- Random Access Memory (RAM) slack</li> <li>- Disk Drive slack</li> <li>- File slack</li> </ul> <p>Swap files</p> <p>Unallocated Space</p> <p>Partial Files</p> <p>Windows Artefacts</p>	While there are some differences, Windows and Linux file recovery is basically similar.
Hidden Files	Files can be assigned an attribute so that they will not be displayed with normal file system viewing methods. Files may be marked as hidden to protect files from being corrupted by casual users or to hide illicit data or activities.	



### 2.4.2 Volatile Data

According to Carvey (2004), “Volatile data ceases to exist when power is removed from the system (either by unplugging the system or shutting it down), and generally includes (but is not limited to) information regarding system processes and applications or services running on the system, network connections to and from the system, and the contents of the system clipboard”.

According to Bejtlich et al. (2005), “analysing volatile data of a victim computer usually contains significant information that helps determine the ‘who’, ‘how’ and possibly ‘why’ of the incident”. This is becoming increasingly important due to the expanding size of volatile memory contained in many computer systems. At the time of writing, desktop systems commonly have 4 Gigabytes of RAM while high end systems may have 16, 32 or even 64 GB (Dell, 2010). Table 2.2 shows the volatile data types and avenues contained in the computer system.

Table 2.2: Volatile Data Types  
(in Carvey, 2004; Bejtlich et al., 2005; and Nolan et al., 2005)

Volatile Data Types and Avenues	
System date and time	
Logged on user(s)	
Process information	
Network connections	
Open Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) ports	
Control Protocol (TCP) or User Datagram	
Internal routing table	
Network status	
Clipboard contents	
Command history	
Services/driver information	
Scheduled jobs	
Open files	
	Random Access Memory (RAM)
	Registers
	Cache

### 2.4.3 Digital Evidence

Digital evidence itself needs to be clearly defined. According to Craiger et al. (2005), “*In 1999, the Scientific Working Group on Digital Evidence (SWGDE) defined digital evidence as: Information of probative value stored or transmitted in binary form*”. Digital evidence is electronic in nature and can be found as data on computer systems that can refer to documents or events that occur within a computer system or network and also on media. In line with this definition, examples of digital evidence could be common application files (e.g. word processing), graphical files, audio and video recording files, server logs, and application executables.

Additionally, operating systems and computer programs also store digital evidence in a variety of places and at a variety of levels. This is seen in Carrier (2005) where data in a volume is analysed to determine where file system and hidden data might reside. The file system can be analysed to find files and recover deleted files. Then a specific file can be analysed to determine what programs were running or to determine content, which may be of evidential value, for instance, the picture contained in a JPEG image file. Any digital evidence that has been recovered can be categorised during analysis into one of three major categories of evidence (Carrier, 2002):

- Inculpatory evidence: that which supports evidence of guilt
- Exculpatory evidence: that which supports evidence of innocence
- Evidence of tampering: that which suggests evidence of tampering

Digital evidence may be sought in a wide array of computer-related crimes, and computer forensic examinations use a variety of methods for discovering data that resides in a computer system, or for recovering deleted, encrypted, or damaged file information. Computer crime includes, but is not limited to, the theft of intellectual property (copyright), child pornography, threatening letters and fraud. All of these crimes would leave varying types and degrees of digital material which can be investigated and may be used to link a suspect to the crime.

The ability to access and analyse the content of files is important to the success of an investigation. As a result of different file types and formats of data, and where the data resides, the investigator is faced with the need to be equipped with tools, techniques and approach to the acquisition and processing of digital evidence.

## 2.5 Methodologies, Tools and Techniques

The unique requirements of computer forensics have stimulated the creation of special software designed to either collect relevant data or to analyse that data in order to find information related to a specific case (Kruse II and Heiser, 2002). A number of forensic tools have been created aimed at achieving the different requirements of the investigation methodology outlined in the previous sections and the requirements of these tools and some examples are discussed in this section.

In many cases, information is gathered during a computer forensics investigation that is not typically accessible by the computer user. Therefore, investigators use a variety of techniques and both open source and proprietary forensic applications to examine a forensic copy of the original media. The investigator's objective is to examine all of the available areas on the drive for possible fragments of data; this includes system data, deleted, encrypted, hidden or damaged files (Hailey, 2003; Carrier, 2005).

There are tools and techniques for each stage of the investigation: the identification, extraction, preservation and documentation of computer data. When considering the various types of electronic storage media capable of holding digital information that may be subject to forensic analysis there is a need for a wide range of functionality. The required tool capability can be implemented as a single tool or as a series of specific tools that are used to handle specific data. These tools differ in cost, functionality, and complexity and maybe open source or closed source proprietary tools. Typically,

the majority of tools available can be divided into two categories: those that acquire evidence and those that analyse evidence. Normally, an acquisition tool will contain some internal verification mechanism, to prove that the copy is exact and has not been altered. In addition, a degree of presentation functionality is often included in analysis tools. There are examples of software tools suites which provide most of the functionality required for an investigation: EnCase from Guidance Software, and the Forensic Tool Kit (FTK) from AccessData.

The process of verification to ensure evidence has not been modified is an essential part of many forensic tools, where it is vital for verifying data integrity, providing continuity and assure provenance. This is usually achieved by a hash function, where it is used to verify the integrity of the original media after imaging, and used to verify the integrity of working copies against the original media. This ensures chain of custody, protecting the integrity of the evidence demonstrating that it has not been altered or tampered with while it was in custody. Current best practice for the forensic acquisition process is to make a forensic copy, to capture all of the user addressable portions of the storage device. Problems have arisen due to the possible manipulation of the storage device at a firmware level (Sutherland et al., 2010), and also as a result of dealing with diverse kinds of evidence. The latter problem is being explored with approaches that attempt to standardise the format used to store digital evidence (Garfinkel, 2006; Turner, 2005).

### 2.5.1 Digital Evidence Formats

The evidence collection in digital forensics investigations includes the methods, techniques and procedures used in retrieving evidence (Newman, 2007). To facilitate the analysis and interchange of data, some sort of data organisation attributes must be considered. One aspect is a standard format to digital evidence storage. The Common Digital Evidence Storage Format

Working Group (2006) states *“without standards that are both open and technically sound, the risk is that evidence may be lost, cases may be compromised, and innocent people may be improperly convicted-or guilty parties let free”*.

The most commonly used forensic formats copy a disk drive: one method is for the forensic tool to make an exact copy of the original material by creating a bit-stream data copy of the disk. The Linux / Unix based ‘dd’ image file which copies chunks of data from one file and writes it to another referring to the file systems or file content. This is sometimes referred to as a ‘raw’ image, a sector by sector copy of the data of a device into a file (The Common Digital Evidence Storage Format Working Group, 2006).

An alternative method is to capture evidence not in the form of a raw image but rather as an embedded image format. This is commonly a forensic tool producing a proprietary format by making a bit-stream copy that contains data from the source device with additional descriptive data such as the date and time of capture and associated case details. There may also be one or more hash values or cyclic redundancy checks (CRC) to assure data integrity. Some tools also create a raw image and a separate file is used to save the additional details. S01: the SMART evidence file also uses compression (AccessData, 2009). The EnCase Evidence File Format is a proprietary format that has become the de facto standard for forensic files. The E01 EnCase evidence file format contains the evidential data, possibly compressed and numerous CRC checks that serve to preserve the chain of custody (Guidance Software, 2005). The implementation consists of series of compressed pages of a disk images. These pages can be individually retrieved or searched and decompressed thus allowing random access to the contents of the image file. Besides using other files for additional information, EnCase embed additional information in their files to provide integrity checks.

Related work in digital evidence storage methods and metadata specifically relating to storage methods for digital evidence includes EnCase, (2005); Garfinkel, (2006); and Turner, (2006). Notable is work by Turner on the

use of Digital Evidence Bags for the acquisition and processing of digital evidence obtained from different digital devices and sources. The method relies on the container used to store the captured information. The unification of digital evidence from different sources automatically creates three types of file as shown in Figure 2.2: .tag file, .indexnn file and .bagnn file.

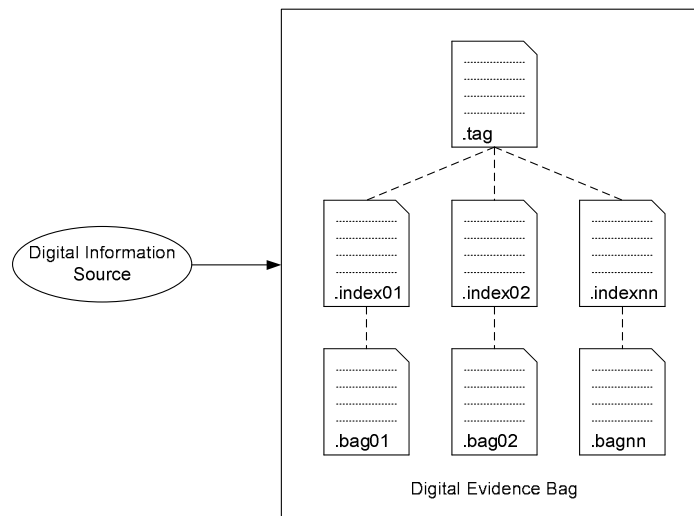


Figure 2.2: A Digital Evidence Bag Comprising Three Files (Turner, 2006)

The additional descriptive data for a bit-stream copy, also referred to as metadata for the hard drive copy, is contained in the plain text file of a tag file. The index file is a text based tab delimited file and detailing the contents of the corresponding bag file, such as folder paths, a list of filenames, and timestamp information. The 'bag' component of the file contains the actual evidence captured in the format of raw binary information, files, structured text or categorised files.

Garfinkel (2006) employed the term Advanced Forensics Format (AFF) to describe an approach where imaged disk storage and compressed data were used to store any type of forensic data, such as disk images and exported files. A series of pages or segments of an imaged hard drive are compressed before

being stored with associated metadata. In addition to laying out information, the AFF creates a variable-length structure called an 'AFF segment' that consists of a header, segment name, flag, data payload area and footer as shown in Figure 2.3. The AFF considers metadata as a separate file, and also within the same AFF file.

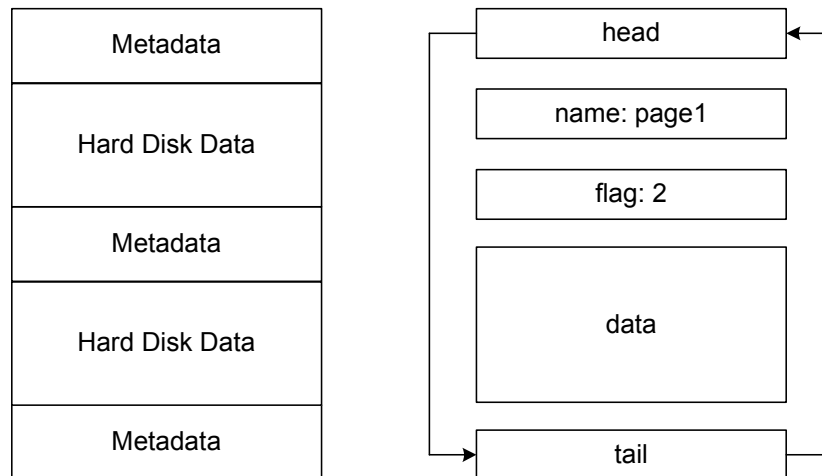


Figure 2.3: An Overview of the Advanced Forensics Format (Garfinkel, 2006)

Similar to the work undertaken by Turner (2006), a study group formed by the Telecommunication Standardisation Sector from the Republic of Korea (International Telecommunication Union, 2009) proposed a common digital evidence exchange file format for communication between digital evidence sites and between different types of forensic tools. Examples of digital evidence sites are: digital evidence extracted from electronic devices, disk image for electronic device, disk image for physical part of electronic device, digital evidence extracted from disk image and digital evidence being transmitted via network. The definition used for disk image is a single file containing the complete contents and structure which represents a data storage device, such as a hard disk, CD, or DVD; and electronic device is any probative information stored or transmitted in digital form that a party to a court case

may use at trial. In this digital evidence exchange file format, as shown in Figure 2.4, a file header, data segment and trailer (containing a hash) are included. The File header includes the information about electronic evidence for exchange and method for integrity check and sender verification. In the data segment, the subject for exchange is included which can be one of the sites of digital evidence. The file trailer is comprised of two kinds of hash value: Data Hash and File Hash for integrity check and File Signature for sender identification.



Figure 2.4: An Overview of the Digital Evidence Exchange File Format  
(International Telecommunication Union, 2009)

While there are a number of methodologies and techniques that can be applied and tools which investigators may use, the tools tend to focus on specific problems and have varying different functionality. For example, ThumbsPlus for displaying images files (Kruse II and Heiser, 2002); SafeBack is primary used for imaging the hard disks (Solomon et al., 2005); and Password Recovery Tool Kit (PRTK) for recovering password (Middleton, 2002). It has been suggested that Richard and Roussev (2006), “*digital forensics tools need to employ more sophisticated data analysis techniques and better collaborative functions to allow digital forensics investigators to perform*



*investigations without becoming overwhelmed by low level details, such as physical disk organisation or the specific file structure".* Therefore, there is a potential for the development of an architecture designed to be extensible and an initial prototype implementation focussed on developing an automated approach for the investigators to sift through System Generated Artefacts. The proposed approach also visualises the System Generated Artefacts in a way that the investigator can easily see what data is available within these areas of the Windows Operating system. This work focuses on the analysis of these artefacts.

### 2.5.2 Tools and Framework

Computer forensics is concerned as much with following prescribed procedures for evidence collection as with the technical aspects of collecting digital evidence. There are several guides that discuss the collection and acquisition of evidence:

- Best Practices for Computer Forensics (SWGDE, 2006)
- Electronic Crime Scene Investigation – A Guide for First Responders (USDOJ, 2001)
- First Responders Guide to Computer Forensics (CERT, 2005)
- Fundamental Computer Investigation Guide for Windows (Microsoft TechNet, 2007)
- Good Practice Guide for Computer-Based Electronic Evidence (ACPO, 2003)

However, these guides are aimed at the most common scenarios, along with these the investigator needs to have skill, techniques and tools. Furthermore, with the ever-expanding role of digital evidence in civil, criminal and employment case, it is not possible to specify guidelines for each type of case. It is necessary for computer forensics organisations and agencies to have

a comprehensive policies and procedures (Volonino et al., 2007).

The need to combat computer crime requires the creation of investigative and forensic tools. Furthermore, recent advancements in computer technologies where capacities at the time of writing area are around 2 terabytes and are estimated to reach 4 terabytes in size by the end of 2011 (Hitachi, 2008), and the advanced techniques for hiding, wiping, encrypting and deleting digital data leave the investigator with an increase in both the number and complexity of cases. They also leave digital forensics in need of tools that are significantly improved, both in richness of features and in speed of operation. Advanced forensic tools are needed to reduce the tedious effort of forensic examiners, especially when searching large hard drives. Richard and Roussev (2006) highlight the need of the digital forensics community for new tools and strategies for the rapid processing of large forensic data sets, which can now originate from numerous sources. Methods for exploring this data including using visualisation techniques help to display information about computer data can help forensic examiners.

Gillam and Rogers (2005) developed a tool that has four basic functions: the ability to search a hard drive for image files; present an interface for an examiner to browse through the images found and select those that are relevant; generate a report of the search function that include the full logical path of the file; and minimal training for the user to utilise the tool. This tool uses recursive directory search, pattern-based search and header-based search to search images in a file. Pattern-based search is done by using asterisks as wildcards that make simple or complex filters possible. Header-based search is used to overcome the problem of file name or file extensions being changed to hide the images. This search is done first by using ImageFormat class in .NET and next by an algorithm that tests the raw format of a file during a search process. This example tool by Gillam and Rogers (2005) highlight the basic functions that need to be performed by a tool when searching a hard drive for

images files, and shows the methods used in creation of investigative and forensic tools.

In the creation of investigative and forensic tools, Vlastos and Patel (2007) focus on the problem of how to visualise digital evidence in an intuitive view, easy and constructive manner for deleted files, wiped files, encrypted and transformed files because of the advanced techniques for hiding, wiping, encrypting and deleting digital data. The tool uses a 3D visualisation technique for displaying the data in the specific block or square of a hard disk drive images. The hard drive's image is split into blocks, partitions and files which have their own view associated with them: the block view, the explorer view and the tree view. Splitting the tool into two smaller modules, one for the extraction of data and the other for the presentation of the data, has proven to be very effective in implementing a visualisation system that offers flexible and extensible capabilities. From this, it has also been learned that XML is more effective for the interchange of data from any forensic tool and storing the data, suggesting this may be a possible technology for implementing a prototype tool in this project.

Besides creating tools for investigative purposes, Bos and Knijff (2005) and Petroni et al. (2006) presented frameworks which can be used as guides to acquire, decode and report evidence. These frameworks work by not containing specific functionality of low-level data extraction. Bos and Knijff (2005) extended the framework concept by making it easy to add functionality without having to think about user interface and repeating common programming tasks. The TUL2G by Bos and Knijff (2005) offers the plug-ins method for adding plug-ins to perform investigation-related tasks without any strict interface to follow through. The FATKit by Petroni et al. (2006) is a framework for system memory's digital forensic data extraction and analysis. The forensic analysis approach of the FATKit is based on system abstractions which are sets of analysis modules that can be added where appropriate.

## 2.6 An Overview of Computer Forensic Analysis and Data Collection

As data can be in many types and contained within different structures, analysis of data can be from many different places as mentioned by Carrier (2005) in Figure 2.5.

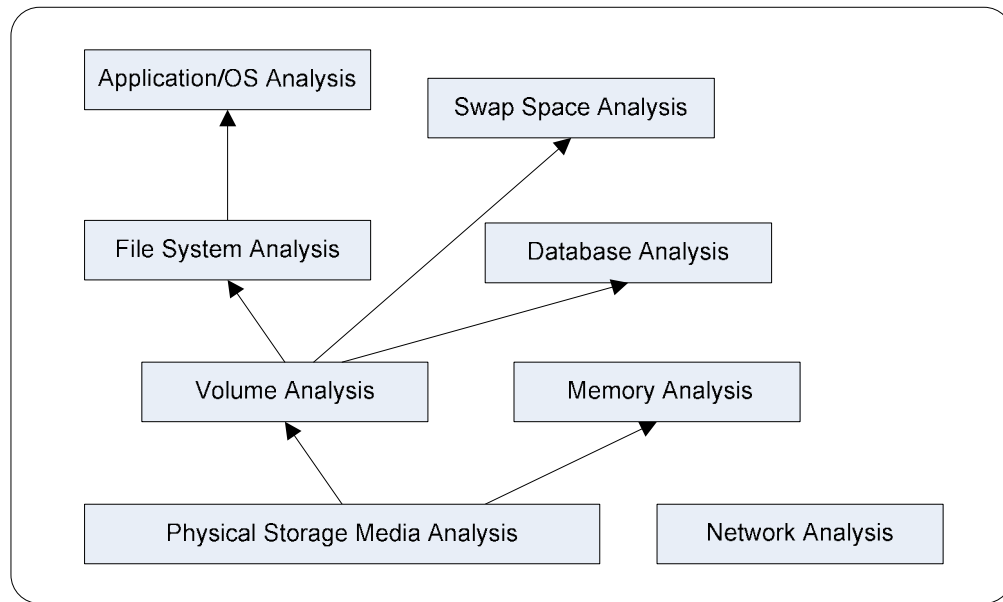


Figure 2.5: Layers of Analysis Based on the Design of Digital Data (in Carrier, 2005)

Carvey (2007) demonstrated that knowing the multiple locations where information is maintained within a particular system, allows an investigator to link information that is found in other areas and to reduce the uncertainty in the analysis. According to Reyes et al. (2007), the analysis phase of the digital forensic process is the point at which the data is explored in more detail and where the data is drawn together and the analysis is the sum of all data applied towards the resolution of the incident. For example, in an intellectual property theft case, the data from a collection of systems were pulled together. The file server audit logs were reviewed and the user list it provided was used

to query the proxy server logs. When the log files for the users were reviewed, a short list was created by focusing on webmail and forum traffic. The short list was used to triage and prioritise the examination of the user workstations, which quickly revealed the individual responsible for the criminal activity when the webmail messages were pulled from the Internet cache, and recreated.

The collection of digital evidence involves preparing the digital evidence to facilitate the analysis stage. The process of collection can be either physical or logical. In physical collection, data is recovered and identified across the entire physical drive without regard to the file system. In logical collection as described by Carrier (2005), data is recovered and identified based on installed operating system(s), file system(s), and/or applications(s) (Figure 2.5). Once the data is extracted, it is analysed, that is to interpret the extracted data to determine its significance to the case.

The following sections explore the collection and analysis process in more detail.

### **2.6.1 Logical Collection**

In the literature (Carrier, 2005), several analysis types of physical storage media to translate various types of usable structure have been found to form the basis for further analysis of physical storage media. We present related work on forensic analysis based on data structure and data organisation as defined in Definitions 2 and 3 in Chapter 1.

Sansurooah (2006) suggests logical collection as an approach for collection based on the installed operating system(s), file system(s) and/or application(s):

- Extraction of the file system information: these are methods that reveal distinctive characteristics such as directory structure; file location, names and attributes; and date and time stamps.

- Data reduction: these are methods to identify and eliminate known files through the comparison of hash values.
- Extraction of files pertinent to the examination: these methods are based on file location, name and extension, and file attributes.
- Recovery of deleted files: these are methods to recover files that have been deleted, which the content of the file remains until the space occupied by the file is re-used by newer files.
- Extraction of password-protected, encrypted, and compressed data: these are methods used to perform an in-depth analysis of a case which is possible through finding the backup copy of the file which is not password-protected, and finding the cleartext versions of encrypted documents, and finding the cache cleartext password, and brute force attacks for passwords.
- Recovering data from the unallocated storage space. This unallocated storage space exists from the end of a file to the end of the cluster assigned to a file which has not been intentionally manipulated by suspects.
- Data recovery of the unallocated space: these are methods to extract data based on when files are erased or deleted. The content of the file is not actually erased only referenced to the data within the File Allocation Table (FAT) that is actually deleted.

### 2.6.2 Physical Collection

It is fair to say that most existing forensic analysis identifies and recovers files and data based on installed operating system(s), file system(s) and application(s). For example, the *ils* tool from the Sleuthkit first recovers the boot sector, then calculates the start of the data region, and finally iterates over chunks of data in the data region when run against a FAT file system (Murr, 2007). An alternative is to identify and recover data across the entire physical drive without regard to the file system. Sansurooah (2006) suggests that approaches undertaken in physical collection use the following methods:

- Keyword searching: this method searches for text or hexadecimal values on the disk. Files including specific words may be of interest and can be found through keyword searching.
- File carving: this method extracts a collection of data from a larger binary object where file system structures are not used during digital investigations, thus, recovers a file from unstructured digital forensic images. This technique carves files from unallocated space using type-specific information, such as footers, headers, and internal structures. An example of this type of tool is the Scalpel (Richard and Roussev, 2005).
- Examining the partition table: this method locates the partition tables and processes them to identify file system structures, including where the file system starts and ends.
- Examining the unused space: this method checks and determines where else evidence can be located in each partition. Two categories of checks can be performed. The first check looks at the last partition and compares its ending location with the end of its parent volume, and if the final partition ends before the end of the volume, there are sectors that can contain hidden data. The next checks compare the start and end sectors of consecutive partitions. If the second partition does not start

immediately following the first partition, the non-partitioned sectors can have been used to hide data and should be analysed.

Nevertheless, only work undertaken using extraction of files pertinent to the physical collection method is elaborated upon in this thesis.

### **2.6.3 Analysis of Data**

The analysis of data refers to the interpretation of the recovered data and placement of it in a logical and useful format, and the correlation and corroboration of possible evidence. Hence, the analysis phase is the phase where the acquired data may be described as 'of evidential value' (Sansurooah, 2006). Digital evidence can in addition to identifying the object and its source, be used to sequence events, determine locations and paths, and establish the time of the action. Analysis of extracted data leads to a more complete picture of a crime – what happened, who caused the events, when, where, how, and why.

In Volonino et al. (2007), approaches undertaken in performing the analysis of extracted data are categorised as: timeframe analysis; data hiding analysis; file analysis; and application analysis. The timeframe analysis method is useful in determining when events happened on a computer system, which can be used as a part of associating usage of the computer to an individual at the time the events occurred. Files are associated with file attributes such as creation date and time, modified date and time, accessed date and time for information as to when the file was used. A user is represented by username and password to log into a system. User and files are used to link the suspect and the data by producing the timeline of named files created, modified or accessed.

Data hiding analysis is useful in detecting and recovering hidden data, which may indicate knowledge, ownership or intent of the data. Files contain information and provide perception of the capability of the system, and the knowledge of the user. This is therefore, an analysis that requires further



steps to be taken when involve file in file analysis. Application analysis is useful when a program exists without the data file on the storage device or a file has no apparent application associated with it.

These four methods can also be categorised into one of three categories of analysis: relational, functional and temporal (Casey and Turvey, 2004). Temporal analysis is conducted to create a chronological list of events by sorting events and actions in the order that they have taken place. This is done with the help of date-time stamps indicating creation, modification and access times to files and folders. Functional analysis is performed to understand how a particular application or system works and to comprehend the meaning of the data it creates. This information can be used to determine what has happened based on the knowledge of how the application responds to a specific event. It is often useful to consider what conditions were necessary for certain aspects of the crime to be possible. Relational (or link) analysis is done in an effort to identify relationships between suspects, victim, and crime scene. Relational analysis can provide information about the geographical locations of suspects, victims, computers and the interactions that have taken place. Determining where an object or person was in relation to other objects or people is very useful when investigating crimes involving networked computers. Relational evidence also includes locations of files and folders on a computer, and location of hidden and missing data.

These categories of analysis include all the methods mentioned above for performing the analysis of extracted data. Timeframe analysis correlates to temporal analysis; data hiding analysis and application analysis and file analysis correlates to relational analysis; and application analysis, file analysis correlates to functional analysis. Correlation between types of analysis is shown diagrammatically below in Figure 2.6.

	Analysis of Data			
	timeframe analysis	data hiding analysis	file analysis	application analysis
Volonino et al. (2007)	↔	↔	↔	↔
Casey and Turvey (2004)	↔		← Functional Analysis →	
		← Relational Analysis →		

Figure 2.6: An Overview of Correlation Between Types of Analysis

The categories of analysis of data that were described in the previous section are further elaborated in the next section. These example works highlight the type of analysis used, methods, and terminologies followed for each author.

### 2.6.3.1 Timeframe or Temporal Analysis

Casey (2004) discusses the need to create a time line of events to identify patterns and gaps, shedding light on a crime, and leading to other sources of evidence. Hosmer (2002) states *“The time line of ‘computer events’ may provide a critical piece of information relating to the prosecution of involved persons. This information can help to pinpoint the location of certain individuals, can assist with the determination of alibis, can uncover conversations and correspondences, and can possibly help to ultimately determine the guilt or innocence of those facing criminal charges”*. The research that has been conducted on the specific use of applying timeframe analysis to digital evidence includes the work by Harms (2006); Murphey (2007); and Kahvedzic and Kechadi (2008).

Harms (2006) considers a timeframe analysis approach when seeking to understand how a computer system had been compromised and for solving an intrusion case through reviewing System Restore points in Windows XP. The approach relied on the time line creation of events on a compromised laptop which exposed how the system had been exploited, subsequent key logger

installation, key logger application execution time, and information on key logger application uninstall date. The timeframe of the attack has been identified as a method of analysis which has answered the questions about this incident. This highlights the value of performing a time frame analysis on system generated artefacts.

A work conducted by Murphey (2007), suggests that the idea of using time frame analysis in correlating events in Windows shortcut (.lnk file) with events in event logs. The event logs in this study were recovered in unallocated space. It was achieved by identifying events during the period of interest and showing specific operation of specific services in an event logs. The attributes of a .lnk file and to be more specific the timestamps are used to correlate the shortcut file and the time series of the event logs. This time series of events illustrates the motivation for timeframe analysis, and even can be viewed as a requirement of forensic analysis toolkits.

In the work undertaken by Kahvedzic and Kechadi (2008), a System Restore Point in Microsoft Windows that archives the Registry and stored the log of changes that happened to the system is used in the analysis of digital evidence. Past user activity in the Windows Restore point have been extracted by comparing multiple Registry hives found within the Restore points. It is done by extracting a `Most Recently Used (MRU)` key from the Registry, comparing it across different Restore Points and extracting the user activity that the MRUs held. The MRU is the list that stores evidence of files names, applications and other information that has been opened by the user in the previous. Each MRU lists particular user activity and updates its content if this activity occurs. RPCompare is one tool developed to address the issues raised in comparing Restore Points (Kahvedzic and Kechadi, 2010). The RPCompare tool was executed on the `OpenSaveMRU` key to extract user activity and a timeline of the different timestamps of this MRU was created. This timeline shows where peaks that indicate a higher amount of new MRU

entries were created and therefore means more user activity during the peaks. This category of methods in analysis of data is shown in Figure 2.7.

### 2.6.3.2 Data Hiding Analysis

Data hiding is the technique whereby a file is changed or manipulated in an attempt to conceal information from the examiner. Simple examples include: changing the extension of a filename; setting a file's attribute to hidden; or using encryption to keep the contents secret (Nelson et al., 2004). There is a body of research investigating techniques that can aid in identifying and assisting in the processing of digital evidence. One such technique attempts to recover data hiding or ways data is hidden is by changing the applicable file extension to hide the file. For example, child pornographers may hide pornographic images by designating them as text files, .JPG or .TXT by simply changing their names (Britz, 2004). Methods employed to hide data are described in Table 2.3.

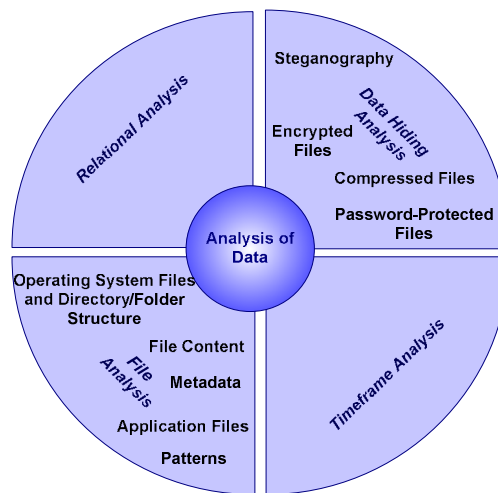


Figure 2.7: Methods for Analysing Data

Table 2.3: Data Hiding Methods

Methods	Descriptions
Password Protected Files	These are methods that depend on a password the suspect uses and prove the suspect intended to keep the contents of a file from everyone else.
Compressed Files	These are methods that depend on file compression used to hide information and make it unreadable by anything other than the compression utility used to save space. An algorithm used to compress a file is used to reverse the compression process to get the uncompressed version of the file. The main reason for file compression is to save a storage device or space in transmission across the networks, but in computer forensics, file compression can also be used to hide information (Volonino et al., 2007).
Encrypted Files	These are methods that depend on encryption used to hide data. In this context, encryption is a method used to conceal incriminating evidence. Files are changed to the point that they are unreadable until the software used to encrypt is used to reverse the process. The most commonly recognised use of encryption is to maintain the confidentiality of information, but if criminals use encryption to conceal their behaviour, an investigator may well be confronted with the data encrypted by them.
Steganography	These are methods that depend on the ability to hide data within another file. This is done by substituting or replacing a small portion of the existing file with the embedded or hidden file or hidden message. This embedded or hidden file or hidden message can be hidden in a sound file, a graphic file, or on unused spaces on a hard disk. The goal of steganography is to avoid drawing attention the transmission of a hidden message. For example, someone who saves pornography to their hard drives may choose to hide the evidence through this method (Solomon et al., 2005).

Criminals can use encryption to prevent access to incriminating data: when they need to obfuscate their illegal activity, they encrypt their communications and the files they exchange. For example, the possible use of Skype that encrypts communication between criminals (Simon and Slay, 2010).

In the work undertaken by Casey (2002), three approaches were involved in recovering encrypted files. First, the examiner needed to find the unencrypted copies of the data that existed before the data was encrypted. To find this data, searches were conducted by the examiner in the disk or from the RAM. This is a further example why system generated artefacts are so important in forensic analysis. Another approach to gaining access to encrypted data was to obtain the passphrase that protected the private key. This passphrase can be obtained from the computer using memory dumps that disclose information relating to encryption or using a systematic method that generates a list of keywords found on the disk. If the passphrase cannot be obtained from the computer, the investigator may be able to obtain the decryption passphrase by searching for slips of paper containing the passphrase or interviewing and persuading the suspect to cooperate. There are various examples of legislation that can be applied to legally require a suspect to surrender encryption keys. An example from the United Kingdom (UK) is The Regulation of Investigatory Powers Act 2000 (RIPA, 2000). This legislation also sets out the legal framework for another approach to obtain the passphrase, that is, to monitor the suspect's machine using software or hardware in an effort to obtain the desired passphrase. The software approach works by enabling key logging, remote file access, and screen captures, thereby helping the investigator to obtain encrypted files remotely. The hardware approach works by recording keystrokes into its

internal memory when the hardware is connected between the keyboard and the CPU.

The third approach in recovering encrypted files was to guess the passphrase used to protect a private key. Manual passphrase guessing is only suitable for a small number of passphrases used is likely, and is feasible to all types of permutations. The automated approach used: a list of common passphrases, a dictionary in the language(s) of the suspect; and more sophisticated permutation techniques (Casey, 2002).

### **2.6.3.3 File Analysis**

File analysis approaches utilize the methods described in Table 2.4. This category of analysis of data is shown diagrammatically above in Figure 2.7.

A possible sequence of activity that took place on the system is the result of a detailed file analysis. It may also possibly result in locating more evidence. Research that has been conducted on the specific use of applying application and file analysis to digital evidence includes that conducted by Gillam and Rogers (2005); Mee et al. (2006); Turnbull et al. (2006); Dongen (2007); Murphey (2007); and Vlastos and Patel (2007).

Table 2.4: File Analysis Methods

Category of Methods	Descriptions
File content	These are methods that depend on the content of a file where obvious files are examined for evidentiary content.
Metadata	These are methods that depend on the metadata information. This metadata information is used by the authority to support evidence which links the user to the action that he/she has taken.
Application files	These are methods that depend on the existence of an application without the data file on the storage device or a file that has no apparent application associated with it. When either scenario exists, this indicates that there may be offsite storage or another storage device being used that contains data files or application software.
Operating system file type and directory/folder structure	These are methods that depend on standard file format and standard folder or directory to store files. For example, the Microsoft operating system has a directory structure different from that of the directory structure of a Macintosh operating system.
Patterns	These are methods that depend on human habits or patterns particular to the suspect. The way files are named or saved as well as time and date indicates patterns.
User configurations	These are methods that depend on what the user has customised on their computer system that leads to finding evidence. For example, questions regarding the configuration that should be asked are: 'Does the user password expire?', 'Is there a remote login feature enabled?' and 'What kind of network settings are in place?'.



In the work undertaken by Turnbull et al. (2006), Google Desktop was used as the source of file analysis in order to find digital evidence. Google Desktop is an application that provides a searching utility on a single Windows desktop computer. As Google Desktop is a searching utility program, the storage files of the Google Desktop hold local emails and store remote emails, HTML Internet pages visited, thumbnail images and certain cached file types such as text. Unique features of Google Desktop are that it caches, indexes and stores Internet sites visited in much the same way as Windows. All the information in the storage files of the Google Desktop can provide information to the forensic investigator during file analysis. Although the storage files of Google Desktop are not humanly readable format, the data that is stored in the storage files of Google Desktop is still accessible.

Forensic analysis of file content produced by Google Desktop is limited to the Google Desktop user interface. The forensic analysis of metadata produced by Google Desktop may not provide an accurate representation of the files contained in the machine in which it is situated. The cause of this is that Google Desktop is meant for indexing and retrieving user created data and not indexing all files on a machine. Google Desktop does not search or index all files used to operate and maintain the machine, but search space that are more liable to contain documents stored by the user. However, Google Desktop interface is provided with browse timeline functionality. This timeline functionality allows a user to view the times at which files were opened and cached by the system. This is useful since it can provide a timeline of events internally rather than having to do time frame analysis separately.

The limitation of most concerns to Turnbull et al. (2006) is that most files created and used by Google Desktop are not humanly readable format, and this format is not known to the researcher. However, the data of the Google Desktop can be viewed by using the Google Desktop program

itself. But, this is not an optimal solution because the data has been filtered and contaminated by the Google Desktop program. Similar to the work undertaken by Mee et al. (2006), and Turnbull et al. (2006) used the Registry as the source of file analysis in order to find evidence for Internet usage. According to Mee et al. (2006), *“The Registry is one of the main places to view various pieces of information relating to the operating system, applications that have been installed on the machine, and information about users who have access to the machine, their settings and the privileges they have to the applications and networks. For example, Microsoft Network (MSN) Messenger stores a cache of contacts for users”*.

The value of the Registry as a forensic artefact through the information it contains includes Websites viewed, network storage accessed, newsgroup accessed, list of users' contacts for instant messenger programs and the Telnet program's lists of recently accessed systems. Registry files are stored in a number of different locations within the Windows system. These files are known as hives, and contain binary format data in groups of keys, subkeys and values. The same as Google Desktop, the Registry data are examined using the Registry viewer for the hierarchical structure of its hives.

Building on the work undertaken by Dickson (2006), and Dongen (2007) provided a further use of file analysis to examine artefacts left by Windows Live Messenger 8.0. There are eight types of artefacts which are left behind after the use of Windows Live Messenger 8.0 on Microsoft Windows XP: artefacts to identify which Windows Live Messenger (WLM) accounts have been used on the computer (checking Windows application event file – AppEvent.Evt); artefacts to show where the contact files of WLM accounts can be found and what useful information they contain (searching Member.stg, .WindowsLiveContact and .CONTACT file); conversation content and the

condition in which it can be found (swap file, hyberfil.sys, Master File Table file); IP addresses for file transfers between sender and receiver; chat/message log files; traces of shared/transmitted files with a contact; audio and video such as voice clips and webcam sessions; and contact and user display pictures. Therefore, by analysing all of these traces it is possible to obtain an overall picture of a user's WLM activities.

#### **2.6.3.4 Relational or Link Analysis**

Expanding on work in several areas of intrusion detection systems, attribution techniques and alert correlation, Wang and Daniels (2006) used relational analysis to identify members of an attack group and their relationship. Attackers, victims, stepping stones and background attackers are the most common members of an attack group. Through the attack group members' identification and their relational links, Wang and Daniels (2006) contributed to network analysis by proposing a novel graph model and hierarchical reasoning framework. This category of methods in analysis of data is shown diagrammatically above in Figure 2.7.

## **2.7 Conclusion**

Most of the work undertaken in forensic analysis focuses on identifying and employing data from a file system to be used as digital evidence. This is due to the common practice of investigators viewing digital data objects that have content or substances that can be perceived. It has been demonstrated by Gillam and Rogers (2005); Mee et al. (2006); Turnbull et al. (2006); Dongen (2007); Murphey (2007); and Vlastos and Patel (2007) that the use of

data extracted from the file system is beneficial in representing the significance of the file as forensic object. Nevertheless, such an approach may not be applicable to files that are not well-documented. Therefore, we need to include additional files that are not categorised as within the file system, as a forensic object. An example of such a file is the swap file, which is a disk-based file controlled by the Memory Manager (The NT Insider, 1998). Thus, we need to include provisions for analysing, searching and correlating other files without expecting the user to know the data structure and attributes of those files, i.e. little will be known about a given file. While many approaches have only a limited view of the file, nevertheless visual approaches that aid the interpretation process can be realised.

Existing analysis of a file system uses a file viewer or editor to display the file contents. This means that a user has to obtain and access an appropriate viewer or editor to display and view the file contents. However, since the viewer or editor is the interface by which the user primarily views and navigates through the file, there is a need to have a forensic analysis system that includes visualisation in assisting investigators to interpret data. This research differs from that in previous studies in that we are interested in analysing forensic objects by integrating data from different sources and in developing an interface capable of visualising file contents, file statistics and file information. We see the limitations of the existing viewers or editors to display the file contents, as a log viewer such as the Event Viewer only interprets the saved log if the type of log file is correctly set and the Registry Editor provides support only for keys and values of a live Registry logical structure (Anson and Bunting, 2007).

My intention is to extend the analysis process supported by such file viewers or editors by including visualisation to represent information in a file.

To gain insight a file's content, analytical visualisation is employed, which enables us to scrutinise large numbers of data. Information on data structure, and a good understanding of the priority of the requirements to manipulate and visualise are inferred from the research of the system generated artefacts and the event logs and swap files be the focus, review of the state-of-the-art tools in examining these system generated artefacts; and later employed as requirements of the development of a flexible and extensible architecture to process the various system generated artefacts.

## CHAPTER 3

# SYSTEM GENERATED ARTEFACTS AS FORENSIC OBJECTS

This thesis is focused on the evidence contained in Windows system generated artefacts. As such the following chapter provides a general introduction to some of the artefacts generated by the Windows operating system. It discusses a number of traces which are left behind after the use of a Windows system which motivate further analysis of the artefacts and possible tools that can be applied to extract information of evidential value. Chapter 2 presented the background of computer forensic analysis, and discussed the theory and practice of examining artefacts extracted from the Windows operating system. This chapter will describe state-of-the-art tools available to the digital investigator. These are also examined and discussed including limitations and possible areas for improvement, and the generation of new forensic tools to analyse system generated artefacts are covered accordingly in this chapter. The intention is to demonstrate the need for an architecture that will specifically extract the information highlighted in the following sections of this chapter.

### 3.1 Introduction

In searching a digital crime scene for evidence, evidence can be recovered from various locations, most commonly from within a volume or file system. The emphasis is often placed on files and their content which involves numerous searching methods such as file names or naming patterns, keyword searches of file content or searching files based on the metadata such as the last accessed or written time listed. According to Carrier (2005), *“The result of file system analysis can be file content, data fragments and metadata associated with files”*.

Volonino et al. (2007) states that *“system generated artefacts are files methodically created by the operating system, such as the metadata, duplicate pointer files, link files, swap files, event logs, and temporary data/cache files. The users do not create these files, therefore contained valuable evidence. The fact that the files are not readily visible to the casual user also makes it more plausible that these files have not been altered”*.

Richard and Roussev (2006) describe *“a forensic process model as follows: for each file in a given file system, perform a number of type-specific operations such as indexing, keyword searches, thumbnail generation, and others. Digital evidence, such as deleted files, file slack, directory structures, registries, and other operating system structures (which include system generated artefacts) are treated and represented as special file types in the forensic process model”*.

## 3.2 System Generated Artefacts

There are a number of possible files which can provide sources of digital evidence which have been discussed in previous sections, including hidden files, file recovery, web artefacts, temporary files and system files. However, some of these files are created by the user and so the user may be aware of their existence and may make some effort to remove them. These system artefacts, which are created by the operating system, are important for digital investigators as they capture a user's activities and are often overlooked by users or intruders as they attempt to cover their tracks. These artefacts are normally hidden from the normal user and often require specific knowledge or specialised tools to find it and access the information. For example, assume a suspect is working on a spreadsheet and wants to check her e-mail. She does not close the spreadsheet window but instead opens a new window to read the e-mail. In order to free up RAM, the operating system places memory being used by the inactive window (spreadsheet) in the pagefile and then proceeds to address the active window. If the user goes back and forth between the spreadsheet and e-mail, valuable information is stored in the pagefile from both the spreadsheet and the e-mail program. Since users have so many generates for unrealised files (in above example, spreadsheet file and e-mail file created in the pagefile), it becomes necessary for the digital investigator to focus on these system generated artefacts' evidentiary values.

Based on existing work, system generated artefacts can play a significant role in aiding a digital investigation and when searching for evidence (Casey, 2000; Jones, 2003; Mandia et al., 2003; Anson and Bunting, 2007; Carvey, 2007; Murphey, 2007) since they contain information concerning the activities that occur on a Windows system and may contain significant evidence available in a digital investigation due to the fact that specialised tools are needed to access them. In order to identify the values of various



artefacts, it is necessary to understand the information contained in these files and the various files' internal structures. It should be noted that in some cases it may be clear how the files are structured due to metadata included in the files, in other cases the data may appear to be unstructured.

Considering extant work by Casey (2000); Jones (2003); Mandia et al. (2003); Anson and Bunting (2007); Carvey (2007); and Murphey (2007), this project examines the six most commonly considered artefacts: Event Logs, Swap File, Registry, Cookie Files, Recycle Bin and Internet Explorer Activity Files. Table 3.1 provides a brief description of some of these artefacts.

Table 3.1: System Generated Artefacts

Windows Artefacts	Brief Description
Event Logs	Event log files record information about which users have been accessing specific files, successfully logged onto a system, unsuccessfully attempted to log on to a system, track usage of specific applications, track alterations to the audit policy, and track changes to user permissions (Mandia et al., 2003).
Swap File	A swap file is a disk-based file controlled by the Memory Manager (The NT Insider, 1998).
Registry	<i>"A central hierarchal database in the Microsoft operating system that maintains configuration settings for applications, hardware devices and users"</i> (Carvey, 2007).
Web Cookies	A text file containing information about web sessions. This is placed by the web server on a user's computer so the web page may be requested back at a later date (Jones, 2003).
Recycle Bin	A file containing files marked as deleted on a Windows system. A file that has been deleted by mistake may be retrieved provided the Recycle Bin has not been emptied (Casey, 2000).
Internet Explorer Activity File	A Web browser (Firefox, Internet Explorer etc.) caches the content of visited web pages and cookies within system files named index.dat (Jones, 2003).
Prefetch	A Prefetch cache to speed up boot and application launch time. Prefetch caches take information from the boot process and from Scheduled Tasks (Hay, 2005).

The artefacts of Microsoft Windows listed in Table 3.1 contain significant amounts of digital evidence that enable investigators to reconstruct activities that took place on a machine before it was seized. This chapter focuses on the Event logs and the Swap file, although it examines the usefulness of registry, web cookies, recycle bin and Internet explorer activity file in the forensic analysis of a system. The next sections consecutively discuss the features, evidentiary values, tools and issues related to event logs, swap files, registry, web cookies, recycle bin and Internet explorer activity file as sources for evidence collection.

### **3.2.1 Event Logs As System Generated Artefacts**

The Windows operating system creates event logs in the process of recording day-to-day events that occur on a Windows system. This is performed by the Windows service named eventlog. The service starts when Windows loads on all platforms by default (Allen, 2005).

#### **3.2.1.1 Event Logs Features**

The Windows event logs files are, essentially, databases with the events related to the system, security, and applications. These events are audited and written to one of three configurable event log files: AppEvent.Evt, SysEvent.Evt and SecEvent.Evt. These three files are stored in the SystemRoot folder of the system. Table 3.2 below illustrates the different default locations in which the various versions of Windows store the .Evt file.

Table 3.2: Event Logs Organisation

Windows Artefacts	Brief Description, Windows Version and Location	
Application Event Log	Contains a log of application usage and logged messages from the operating system and programs.	
AppEvent.Evt	Windows NT 4.0	%SYSTEMROOT%\system32\config\
	Windows 2000	%SYSTEMROOT%\WINNT\config\
	Windows XP	%SYSTEMROOT%\system32\config\
	Windows Vista	%SYSTEMROOT%\system32\winevt\Logs\
Security Event Log	Records activities that have security implications such as logins.	
SecEvent.Evt	Windows NT 4.0	%SYSTEMROOT%\system32\config\
	Windows 2000	%SYSTEMROOT%\WINNT\config\
	Windows XP	%SYSTEMROOT%\system32\config\
	Windows Vista	%SYSTEMROOT%\system32\winevt\Logs\
System Event Log	Notes system events such as shutdowns.	
SysEvent.Evt	Windows NT 4.0	%SYSTEMROOT%\system32\config\
	Windows 2000	%SYSTEMROOT%\WINNT\config\
	Windows XP	%SYSTEMROOT%\system32\config\
	Windows Vista	%SYSTEMROOT%\system32\winevt\Logs\

The event logs consist of a binary structure, with a header and a series of event records stored in the file. The event log is maintained as a circular buffer since older event records are cycled out of the file whenever a new event record is added to the file. At the same time, there is correlation between the event logs, registry and many message files (DLL) on a system (Carvey, 2007).

The location of event logs is dependent on the version of Windows running on the computer. According to Anson and Bunting (2007), in order to examine the contents of an event log, the event log header and event records contain structure, values and information as shown in Tables 3.3 and 3.4 that can assist an investigator in recognising and interpreting event log files. The event log records contain information about an event such as the date, time, user, computer, event ID, source, type and category (Microsoft, 2007).

Table 3.3: Event Logs Header Structure

Field Number	Offset (bytes )	Size (bytes)	Description
1	0	4	Size of the record; for an .evt file header, the size is 0x30 (48) bytes. Event record sizes are 56 bytes.
2	4	4	Magic Number ("LfLe") 0x654C664C, or 4C664C65 (LfLe) when the endianness is reversed.
3	16	4	Offset within the .evt file of the oldest event record
4	20	4	Offset within the .evt file of the next event record to be written
5	24	4	ID of the next event record
6	28	4	ID of the oldest event record
7	32	4	Maximum size of the .evt file (from the Registry)
8	40	4	Retention time of event records (from the Registry)
9	44	4	Size of the record (repeat of DWORD at offset 0)

Table 3.4: Event Log Records Data Structure

Field Name	Offset (bytes)	Size (bytes)	Description
RecordSize	0	4	Beginning of Record Size Marker (4 bytes – 32 bit little endian integer)
MagicNumber	4	4	Fixed Value Delineator or Object Marker (4 bytes – 0x4C664C65, which is ASCII "LfLe")
RecordNumber	8	4	Record Number (4 bytes – 32 bit little endian integer)
TimeGenerated	12	4	Created time stamp (Unix 32 bit little endian time stamp)
TimeWritten	16	4	Written time stamp (Unix 32 bit little endian time stamp)
EventID	20	2	Event ID (2 bytes – 16 bit little endian integer)
EventType	24	2	Event Type (2 bytes – 16 bit little endian integer value used as an index to return "Event Name") (0x01 = Error; 0x10 = Failure; 0x08 = Success; 0x04 = Information; 0x02 = Warning)

Table 3.4: Event Log Records Data Structure (continued)

Field Name	Offset (bytes)	Size (bytes)	Description
EventTypeName	26	2	String Count (2 bytes – 16 bit little endian integer – describes number of strings in the event record)
EventCategory	28	2	Category (2 bytes – 16 bit little endian integer)
EventCategoryName	30	2	Generated by looking up the associated Event Category number
LastRecordNumber	32	4	Closing record number
OffsetDescription	36	4	String offset; offset to the description strings within this event record
LengthSID	40	4	Length of the user SID; size of the user SID in bytes (if 0, no user SID is provided)
OffsetSID	44	4	Offset to the user SID within this event record
DataLength	48	4	Data length; length of the binary data associated with this event record
OffsetData	52	4	Offset to the data
SourceName	56	-	Source Name (Variable length Unicode text with padding and null terminator 0x0000)
ComputerName		-	Computer Name (Variable length Unicode text with padding and null terminator 0x0000)
SID	Field 15	Field 14	SID of Security Principal or Group (may or may not be present). If 1-5, which is S-1-5 (NT Authority – unique identifier), then full SID follows
	-		If SID follows 14, then this is security authority of SID that follows (0-5)
	-		If SID follows 14, then this is remainder of SID, appearing in 5 sets of 32 bit integers
Message	-	4	Strings – Depending on the number of defined strings (Field 9), strings will be in Unicode and separated by an ending with null terminators 0x0000.
Data	Field 17	Field 16	Data – an optional field used when message is unique (typically containing an offset or value for an error, etc). Data is regular text (non-Unicode) with each string separated by 0x20 (space) and ending with 0x0D0A (carriage return) and null terminator
EndRecord	241	4	End of record size marker (4 bytes – 32 bit endian integer)

Windows registers and records events in the Application log, System log and Security log and each of these logs stores different information based on the type of events. Types of event in the Application and System logs are different from the types of event in the Security log. *Information*, *warning* and *error* are the types of event entries that are recorded in the Application and System logs while *success* and *failure* are event entries that are recorded in the Security log. These types of event are used in an attempt to troubleshoot system anomalies and used with other column fields to find indicators of attack.

As indicated in the above section, there are three categories of event logs within Windows. The Application log contains actions that particular software applications identify as events. For example, the antivirus applications use this log to record when the program gets updated and the Microsoft Security Centre will write a record to this log. The Security log contains detail regarding logon events. Such events logged include successful logon attempts as well as unsuccessful logon attempts. The System log contains events logged by the Windows system component. The System log is used by the operating system to track events such as driver failures or when a system component does not start up correctly. For example, when a Windows service fails to start, details will be found in the system logs (Microsoft, 2005a).

One useful piece of information that can be gained from Event logs is a record of which accounts have been used to access the systems and how these accesses have been made. Such accesses are called logon events (Anson and Bunting, 2007). Logon events are generated and recorded in the Security log and these events grant an account access to a computer's resources. In Windows, each of the different versions of the operating system uses a logon type to indicate different kinds of logon events. A logon type can be classified into one of the nine types. Windows logon types are values to indicate the way in which the account logged on to the system. The values for logon types are shown in Table 3.5 below.

Table 3.5: Values for Logon Types (Anson and Bunting, 2007)

Logon Type	Logon Title	Description
2	Interactive	A user logged on to this computer at the console.
3	Network	A user or computer logged on to this computer from the network.
4	Batch	Batch logon type is used by batch servers, where processes might run on behalf of a user without the user's direct intervention.
5	Service	A service was started by the Service Control Manager.
7	Unlock	This workstation was unlocked.
8	NetworkCleartext	A user logged onto a network and the user password was passed to the authentication package in its plain text form.
9	NewCredentials	A process, thread or program cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity, but it uses different credentials for other network connections.
10	RemoteInteractive	A user logged on to this computer remotely using Terminal Services or a Remote Desktop connection.
11	CachedInteractive	A user logged on to this computer with network credentials that were stored locally on the computer. The domain controller was not contacted to verify the credentials.

### 3.2.1.2 Evidentiary Value of Event Logs

Event logs record and reveal information about activities that occur on a Windows system. These logs can be used to diagnose and troubleshoot problems on a Windows system as the logs record information about hardware and software problems. According to Mandia et al. (2003), by reviewing the System log, Application log and Security log, the following information of possible evidentiary value can be obtained:

- Which users have been accessing specific files?
- Who has been successfully logging on to a system?
- Who has been trying unsuccessfully to log on to a system?

- Usage of specific applications.
- Alterations to the audit policy.
- Changes to user permissions.

Logs may contain event entries with specific information about the recorded event represented by fields that are typically of investigative interest and many activities that happen within a Windows system can be reconstructed by analysing these events. For example, Date, Time, IP addresses and/or Computer Names of involved system fields can be used to determine which computer was used to perform a specific action, what time someone logged into a computer, and from where (Casey, 2000).

An event identifier is assigned to each event whenever an event is audited on the Windows system. The Event ID column contains a number which corresponds to the type of event that has occurred; most commonly associated with logon and authentication activity. An Event ID associated with remote desktop events is useful in identifying the name and IP address of the originating computer for the remote connection. This is used in determining successful remote desktop connection and reconnection, so in some cases Event logs can play an important role in addressing intrusion cases. In cases involving remote desktop connections, Event logs may be the best source of evidence regarding the attacker.

### **3.2.1.3 Event Logs Tools and Related Issues**

Since the logs are stored in proprietary binary format and not in a text-readable format, a special tool is needed to interpret the data and to display the content in a Human Readable form. The contents of the event logs can be viewed using the event viewer tool supplied with the Windows operating system. The Event Viewer tool provided by Microsoft depicts the three classifications of event logs within Windows in two different panes. One pane shows the list of the available log files and the other provides a list of each of



the different event entries, one line per entry. Figure 3.1 shows the two panes of Event Viewer and shows a list of event entries in the Application/Security/System log. Figure 3.2 shows an example of the opened Event Properties dialog whenever a selected entry is double-clicked. There are other tools that include functionality to view event entry by relying on the Windows API, including Log Parser and Event Analyst.

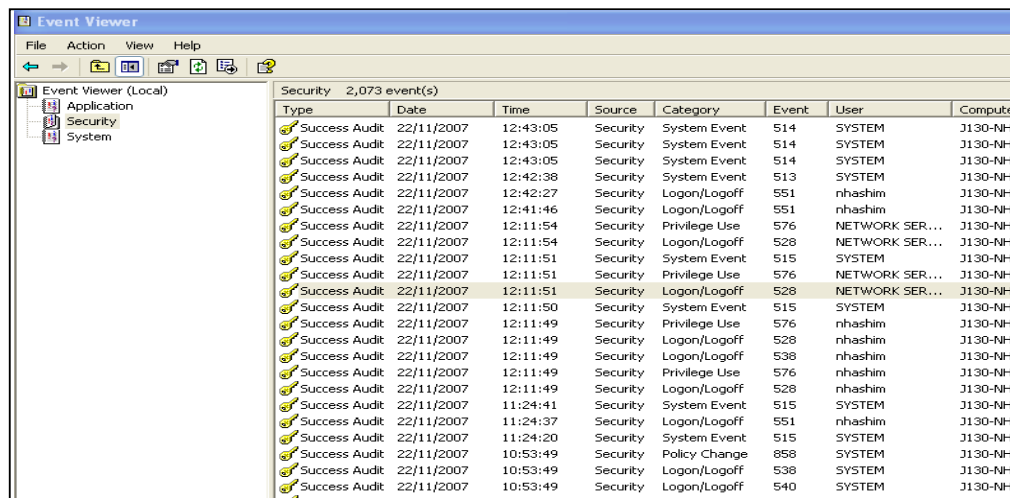


Figure 3.1: Event Viewer Panes

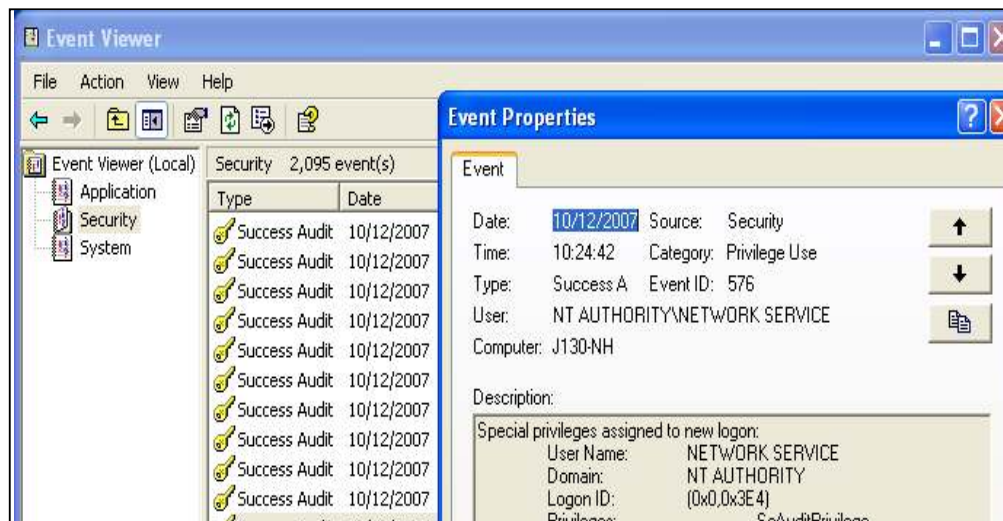


Figure 3.2: An Example of the Opened Event Properties Dialog in Event Viewer

Another free tool provided by Microsoft to process and view event logs is the Microsoft Log Parser (Microsoft, 2005). It can be downloaded from the Microsoft Download Centre<sup>1</sup> and is able to process a number of different types of log format. The Log Parser tool processes Event logs in three parts using different components: input, query and output. Its query component uses SQL queries to parse, filter and analyse logs. The Log Parser uses SQL for the searching function and so it can be difficult for users who are not already familiar with SQL. The input component is the specified input type of interest, while the output component is the defined output type that is wanted to be displayed.

There are commercial tools that perform some basic log analysis functions. For instance, the LANguard Security Event Log Monitor (LANSELM), from GFI (2007) is an event logs monitor that retrieves event logs from networked NT/2000 servers and workstations and alerts the administrator of possible intrusions.

Read (2009) mentions several tools that work with many different log formats for detailed log analysis, such as the Spotfire (TIBCO, 2011) and Sawmill (VisiData, 2011). Such a commercial tool is based on an Intrusion Detection Systems (IDS) and there is a need to undertake more in-depth analysis of event logs, such as fusing or correlating event logs from multiple computers. The ability to identify a set of interesting log events across different systems can be a useful function; and data mining tools can be used to extract a pattern of repeated yet unusual events from the event logs.

One possible issue in examining data from multiple systems in correlation of the data is the question of the reliability of the timestamp. Jeffrey and Clark (2000) noted that event logs store the date that the entry was made and time that each entry was written in the log. Windows saves

---

<sup>1</sup> Log Parser available at:  
<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07>.

time stamps in FILETIME format. FILETIME format is the number of ticks, in 100ns increments, since 00:00 1 Jan, 1601 (Microsoft, 2007). This FILETIME format needs to be translated to Unix time format since the Event logs measure the time in Unix format (Carvey, 2007). Unix time format is the number of seconds since 00:00 1 Jan 1970. The timestamp is stored in GMT and no information about the time zone setting for the computer is recorded. The timestamp therefore has to be adjusted in order to analyse it when considering logs that originated from computer systems in different time zone. This compounds the problem of possible variation between the clocks of various systems and the need for system administrators to resolve these issues to compare and analyse events (Carvey, 2007).

In addition to the timestamp issue, there is also the possibility of data loss. According to Anson and Bunting (2007), data loss happens when an event ID has been updated on a newer version of the operating system, and an older version of the operating system is used to interpret it. This happens when analysis is done using an older version of the operating system and the event logs has been created with the newer version of the operating system. It also happens to the username where a username is represented by the SID when a log from a different computer is analysed instead of the log from the local machine. This occurs because the SID of the log for the different machine is interpreted as a username by querying the list of usernames stored on the local machine.

The evolution of the Windows operating system can also complicate the issue (Anson and Bunting, 2007). The operating system controls the way in which the logs are generated, so the evidence found is the consequence of the version of the operating system used by that victim (Anson and Bunting, 2007). The history of the Windows operating system is shown in Figure 3.3.

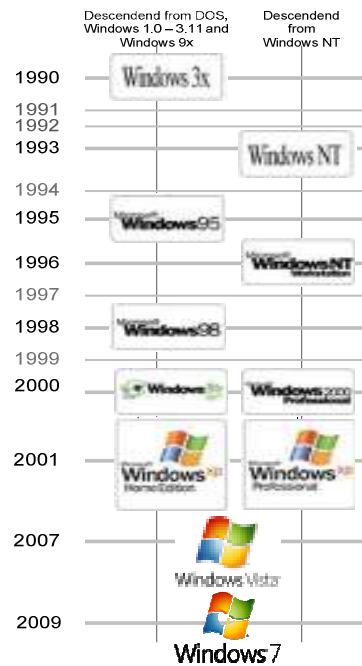


Figure 3.3: Windows Operating System History (Microsoft TechNet, 2007)

### 3.2.2 Swap Files As System Generated Artefacts

According to Lee et al. (2007a), “*System memory analysis aims at gathering information from the contents of a computer’s memory with the purpose of finding which processes were running, when they were started and by whom, what specific activities those processes were doing and the state of active network connections*”. Kornblum (2007) reports that virtual memory implemented by the operating system allows a larger range of memory or storage addresses for stored data than physical memory. The computing system maps the virtual addresses to real hardware storage addresses and also manages storage swapping between physical memory (RAM) and hard disk (swap file).

A swap file can be defined as part of virtual memory and is a disk area where memory pages belonging to various processes can be swapped in or

swapped out, depending on how this is handled this can provide a useful source of activity including material which the user did not intend saving to disk. The intermediate characteristic of a swap file, i.e. containing certain processes from memory but located in a hard disk can be used by a digital forensic investigator to solve the investigation, by using such information in order to acquire critical information such as passwords and credit card numbers that being 'swap out' from main memory to swap file.

A portion of the hard disk is used as additional memory and data is swapped from the physical memory to this space on the disk as memory as needed in the physical memory. This data is held in a file called the Swap file. This file is created automatically by the operating system. Swap files are created by the operating system in a default location.

### 3.2.2.1 Swap Files Features

Caloyannides (2001) described the swap file as the portion of the hard drive used by Windows system to temporarily store data that would normally be stored in the volatile RAM, but the RAM is currently in use by other processes. This file contains all sorts of data, including e-mails, web pages, word processing documents and any other work that has been performed on the computer during the work session (Shinder, 2002). Consequently, the swap file will include residual information; examples include previously opened files and print spooling (Mohay et al., 2003). In other words, a swap file is simply dedicated space on a hard drive whose contents are temporary and overwritten as needed. A swap file is generated at each boot session and remains unless it is configured to be cleaned each time the system is powered down. However, the shutdown period increases when the swap file is configured to be cleared out by setting the registry value to 1 for `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\MemoryManagement\ClerPageFileAtShutdown`.

Table 3.6: Swap File Default Name and Location

File Name	Windows Version	Location
Win386.swp	Windows 95/98/ME	%SystemDrive%\
pagefile.sys	Windows NT/2000/XP	%SystemDrive%\
	Windows Vista	%SystemDrive%\

### 3.2.2.2 Evidentiary Value of Swap Files

When a computer's RAM is full and the operating system must allocate memory for an application that doing some processing, Windows creates the swap space as a single file and this is known as a swap files on the root folder of the system drive to make room in RAM. The default swap file name and the location within the file system is outlined above, Table 3.6. The swap file is a collection of memory pages which belongs to various processes (Lee et al., 2007a), threads (Schuster, 2006) and also stores CrashDump data. This CrashDump data is essentially a dump of the physical memory when a 'blue screen of death' (BSOD) occurs (Ruff, 2007). A commonly used forensic method is the searching of unallocated space for deleted files (Shinder and Tittel, 2002); and this approach can also be applied in searching the swap files for the deleted files and other digital object through the detection of file headers and footers (Casey, 2004). Thus, swap files can include a great deal of information, specifically passwords that were never intended to be recorded onto the hard drive (Lee et al., 2007b), drafts of documents that were never saved to disk, and so on.

Accordingly, there is the possibility of swap files containing a wide range of data including passwords, user IDs, credit card numbers, messenger chat logs and contents of recently used files, such as address books and URLs

(Lee et al., 2007a). An example is displayed in Figure 3.4, where a password is detectable in a pagefile.sys file.

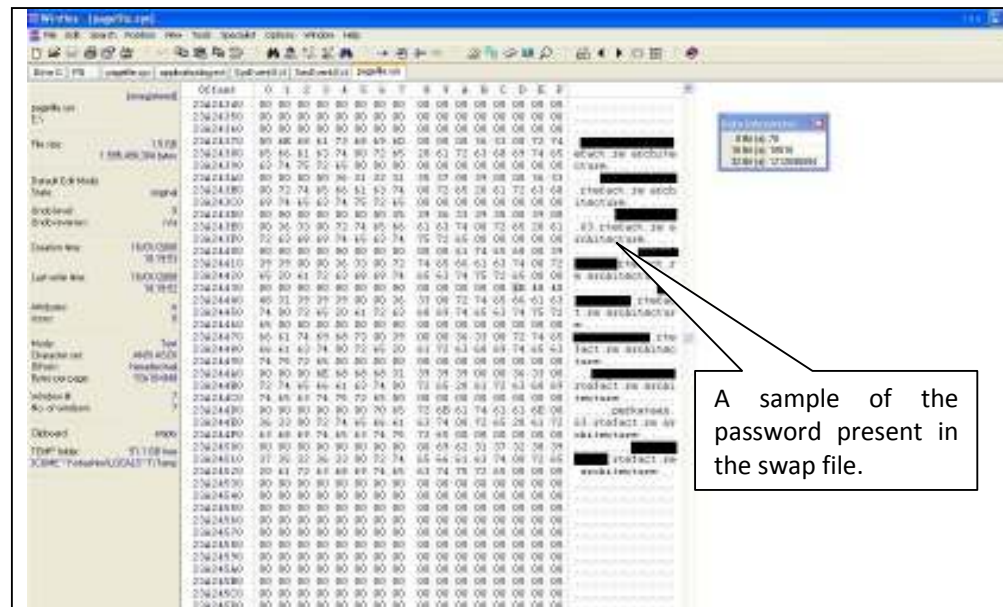


Figure 3.4: Result of pagefile.sys Analysis

### 3.2.2.3 Swap Files Tools and Related Issues

The pagefile is locked by the kernel when Windows system is running. However, the pagefile can be accessed using a specially crafted driver or special device. This special device or crafted driver must be brought onto the target or have been installed previously. Another way to collect the pagefile is to unplug the system and access the pagefile through the standard forensic process hard drive extraction.

An application will fail to operate when the system runs out of virtual memory as a result of insufficient swap file space being provided. Swap file size depends on how much RAM there is, and how much additional memory space that workload requires. For computers with small amounts of RAM (256MB), it is possible to configure the swap file 1.5 times the

size of the installed RAM, but for computers with a large amount of RAM, there is not much point in allocating a swap file that is 8 GB, if the computer has 8 GB of RAM. There is little point in allocating a large minimum size of swap file, since it typically won't be used (Sanderson, 2004). Even if the minimum and maximum swap file size is not the same, fragmentation occurs and this can lead to additional performance degradation. The best way to know the size needed for swap files is to monitor how much of the swap files in use and the associated system paging activity.

The contents of swap files are not easily readable by the investigator. The investigator can view the swap files' contents, but not in a manner that makes the viewing or extraction of valuable data easy. Looking for leads in the swap file by viewing it with standard binary editing utilities is tedious and will most likely be unfruitful because of the volume of data involved. In order to unravel the contents of swap files, more productive, specialised tools are needed so that numerous fragments of page file data can be extracted and assembled (Schweitzer, 2003). There are, however, various computer forensic filters designed to automatically identify computer investigation leads stored in Windows swap files. The identified leads are used to craft lists of key words and strings of text for use with computer forensic tools. Winhex (Vyavhare, 2009) is a freeware tool and is promoted (Volonino et al., 2007) as a tool that can be used to examine swap files in forensic investigations.

### **3.2.3 The Registry As A System Generated Artefact**

The Windows Registry is a database that stores hardware and software configuration information, user preferences, and setup information. It uses a binary format and is optimised to be machine, rather than human readable and so cannot be viewed with a text editor. The Registry has a hierarchical structure similar to the directory structure on the hard disk.



### 3.2.3.1 The Registry Features

The registry is a system-defined binary database designed to store and retrieve configuration data required by applications and operating system components. The binary nature of the Registry means that the registry files cannot be read from a DOS prompt text editor or the Recovery Console and locating the registry files depends on which Windows platform is installed (Russinovich, 1999).

In Windows 95, 98 and ME there are only two registry files, which is the `system.dat` and `user.dat`. In Windows XP, 2000 and Vista there are several registry files. The files are stored in the `Windows\System32\Config` folder. When looking at an offline Registry, the files are: `Software`, `System`, `SAM`, `Security` and `Default` (no file extension is used). One more registry file is `NTuser.dat` and it does have a file extension. Table 3.7 illustrates the registry files from different Windows platforms and their locations. Windows 7 includes further refinements including virtualisation to ensure compatibility with earlier versions. This thesis is however focused on Windows XP and 2000.

Table 3.7: Registry Files From Different Windows Platform and Their Locations

File Name	Windows Version	Location
System.dat	Windows	%SystemRoot%\
User.dat	95/98/ME	
NTUSER.DAT	Windows NT	%SYSTEMROOT%\ System32\Config\ %SYSTEMROOT%\Profiles\ <username>\
system/SECURITY/ SAM/software/default	Windows 2000/XP	%SYSTEMROOT%\System32\Config\ %SYSTEMROOT%\Documentsand Settings\ <username>\
system.alt		
system/SECURITY/ SAM/software/ default.sav		%SYSTEMROOT%\System32\Config\ C:\Users\<username>\
system/SECURITY/ SAM/software/ default .log	Windows Vista	

The Software, System, SAM, Security, Default and NTuser.dat files have various functions. The Security stores information about security. This file contains information regarding various service accounts for the operation of Windows. For example, the system stores the credentials for service accounts and launches them automatically under the appropriate account since services run without being openly activated by a logged in user. The SAM file is a security database of hashed passwords and usernames which stores information about the Security Accounts Manager (SAM) service. For example, each user, group and machine in Windows are assigned a security identifier (SID) and this SID is used as a means of identification. SIDs to users for logon is resolved by parsing the SAM. The System stores all the information about hardware, software, and the default Windows setting. For example, each

service that starts upon Windows booting is listed in the Registry key `HLKM\SYSTEM\CurrentControlSet\Services` and the settings for the Windows Firewall are stored in the Registry key `HLKM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy` in the `SYSTEM` file. The `Software` file stores information about software and how Windows will perform and the default Windows setting. The `HLKM\SOFTWARE` key in this file contains software settings relating to installed software and uninstalled software; and even who last logged on to a system. The `Default` file stores all the default user settings. A number of Registry keys are used to track user activities and can be found in the `NTuser.dat` file and are updated when a user performs specific actions. The `NTuser.dat` file stores all settings that each user selects and these settings will override settings stored in the `System` file.

The structure of the Registry itself contains the keys, subkeys and values at a physical level (in an offline Registry). There are five root level keys. Keys are the master keys (`HKEY_LOCAL_MACHINE` and `HKEY_USERS`) which are two in number and derived keys (`HKEY_CLASSES_ROOT`, `HKEY_CURRENT_USER`, `HKEY_CURRENT_CONFIG`), meaning they are linked to the master keys which are three in numbers. Table 3.8 below shows the different keys, subkeys, and locations of the Windows registry file. Values are the Registry values which are associated with Name, Type and Data attributes. All values have names and value's name is analogous to a file's name. Each value contains data of a specified data type specified by a number. Figure 3.5 shows the Registry editor with a series of values in the value pane.

Table 3.8: Registry Organisation (Anson and Bunting, 2007)

Root Key Name	Brief Description
HKEY_LOCAL_MACHINE (HKLM)	Used to establish the pre-computer settings.
Hive Key	Hive File and Location
HKLM\SAM	%SYSTEMROOT%\system32\config\SAM
HKLM\SECURITY	%SYSTEMROOT%\system32\config\ SECURITY
HKLM\SOFTWARE	%SYSTEMROOT%\system32\config\ SOFTWARE
HKLM\SYSTEM	%SYSTEMROOT%\system32\config\ SYSTEM
HKEY_USER (HKU)	Used to contain the user environment settings for the console user as well as other users who have logged on to the system.
Hive Key	Hive File and Location
HKU\DEFAULT	%SYSTEMROOT%\system32\config\default
HKU\S-1-5-19	Documents and Settings\LocalService \ntuser.dat
HKU\S-1-5-19_Classes	Documents and Settings\LocalService\Local Settings\ Application Data\Microsoft\Windows \UsrClass.dat
HKU\S-1-5-20	Documents and Settings\NetworkService\ ntuser.dat
HKU\S-1-5-20_Classes	Documents and Settings\NetworkService\ Local Settings\Application Data\Microsoft\ Windows\UsrClass.dat
HKU\SID	Documents and Settings\UserName\ ntuser.dat
HKU\SID_Classes	Documents and Settings\UserName\Local Settings\ApplicationData\Microsoft\Windows \UsrClass.dat
HKEY_CLASSES_ROOT	Used to associate file types with programs that open them and also to register classes for Component Object Model (COM) objects. This key is derived from two keys, HKLM\Software\Classes and HKCU\Software \Classes.
HKEY_CURRENT_USER	Used to configure the environment for the console user. This key is derived from a link to HKU\SID. SID is the user's security identifier.
HKEY_CURRENT_CONFIG	Used to establish the current hardware configuration profile.

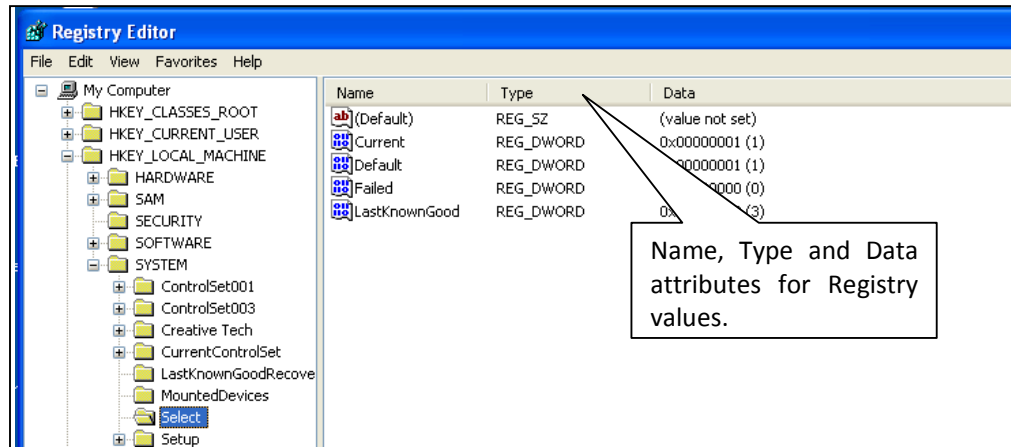


Figure 3.5: Registry Editor Showing Registry Values in the Value Pane

Table 3.9 below shows each of the data types, their corresponding number and a brief description of what the data type means.

Table 3.9: Registry Value Data Types (Anson and Bunting, 2007)

Data Type	Number	Description
REG_NONE	0	Data type is not defined.
REG_SZ	1	Fixed length text string expressed in user friendly format, normally describes components.
REG_EXPAND_SZ	2	Variable length data string.
REG_BINARY	3	Binary data that is displayed in the editor as hex.
REG_DWORD	4	32 bit double word values.
REG_DWORD_LITTLE_ENDIAN	4	32 bit double word values with bytes in reverse order.
REG_DWORD_BIG_ENDIAN	5	32 bit double word values with bytes in normal order with the highest bit appearing first.
REG_LINK	6	An internal use only data type for a Unicode symbolic link.
REG_MULTI_SZ	7	Multiple string fields in which each string is separated by a null; two nulls mark the end of the list of strings.
REG_RESOURCE_LIST	8	Listing of resources lists of devices or device drivers.

The Windows registry is organised in a tree structure and is analogous to a file system. However, the internal structure of Windows registry hives is different from typical file systems. Registry hive files contain a header and continue with a series of hive bin blocks. Hive bins are linked together and within each hive bin can be found a series of variable length cells. Figure 3.6 illustrates the layout of a typical hive bin. The data portion of each cell contains either value data or one of several different record types such as: key (NK) records, subkey-lists, value-lists, value (VK) records, security (SK) records, big data records and big data indirect offset cells (Morgan, 2009).

Figure 3.6 below depicts the Registry hive files contain a header and continue with a series of hive bin blocks and within each hive bin can be found a series of variable length cells.

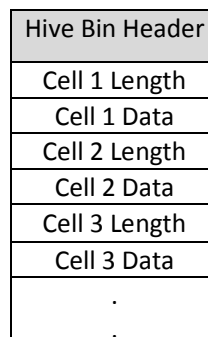


Figure 3.6: Hive Bin Structure (Morgan, 2009)

Table 3.10 through Table 3.12 illustrates data structures for all of the elements in the registry: header, hive bins, and cells. Table 3.10 below depicts the Registry header with the offset values within the file to the key cell.

Table 3.10: Registry Header Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	4	String("ref")	Magic number
0x4	4	Unsigned Integer	Sequence Number 1: matches next field if hive was properly synchronised.
0x8	4	Unsigned Integer	Sequence Number 2: matches previous field if hive was properly synchronised.
0xC	8	Unsigned Integer	64-bit NT time stamp
0x14	4	Unsigned Integer	Major version
0x18	4	Unsigned Integer	Minor version
0x24	4	Offset	Pointer to the first key record
0x28	4	Offset	Pointer to the start of last hive bin in file.
0x30	64	String	Hive file name
0x90	4	Unsigned Integer	Flags
0x1FC	4	Unsigned Integer	Checksum of data to this point in header.

Within the Registry hive file, a series of hive bin blocks that follows after the header contains offsets to additional data structures, such as lists of other hive bins, as well as lists of cells. Table 3.11 below depicts the Registry hive bin data structure with the offset values.

Table 3.11: Hive Bin Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	4	String("hbin")	Magic number
0x4	4	Unsigned Integer	This bin's distance from the first hive bin
0x8	4	Unsigned Integer	This hive bin's size (multiple of 4096)
0xC	16	Unknown	Unknown
0x1C	4	Unsigned Integer	Relative offset of next hive bin
0x20..[bin size]	Variable	Structure List	List of cells used to store various records

The Registry hive file contains data that is stored in cells. A cell holds a key, a value, a security descriptor, a list of subkeys, or a list of key values. Table 3.12 below depicts the Registry cell data structures with the cell header; that is a field that specifies the cell's size, and the cell's value.

Table 3.12: Cell Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	4	Signed Integer	Cell length (including these 4 bytes)
0x4	Variable	Varies	Contains one of: NK record, VK record, SK record, subkey-list, value-list or raw data blocks

Furthermore, the data portion of each cell contains either value data or one of several different record types such as: key (NK) records, subkey-lists, value-lists, value (VK) records, security (SK) records, big data records and big data indirect offset cells (see Appendix B).

### 3.2.3.2 Evidentiary Value of the Registry

The Registry on Windows operating systems is a rich source of evidence and contains data relating to the configuration of the operating system, and most of the applications present on the system. In addition to system configuration data for the operating system and applications present on the system, the Windows Registry holds information regarding user activities including recently accessed files. Software used by attackers creates a footprint within the Registry, leaving the investigator clues about the incident (Carvey, 2007). Due to the binary format of the registry, a specialised tool is required to view the contents. Some, but not all, of the registry information can be accessed using the Registry editor supplied with the operating system. In Figure 3.7, `DefaultUserName` shows who last logged on to a system in the `HKLM\SOFTWARE\Microsoft\WindowsNT\Current\Version\Winlogon\De-`



faultUserName. Similarly Figure 3.7 illustrates another value which can be obtained by looking at the DefaultDomainName in the same key, which describes the local or domain security authority of the last account used to logon.

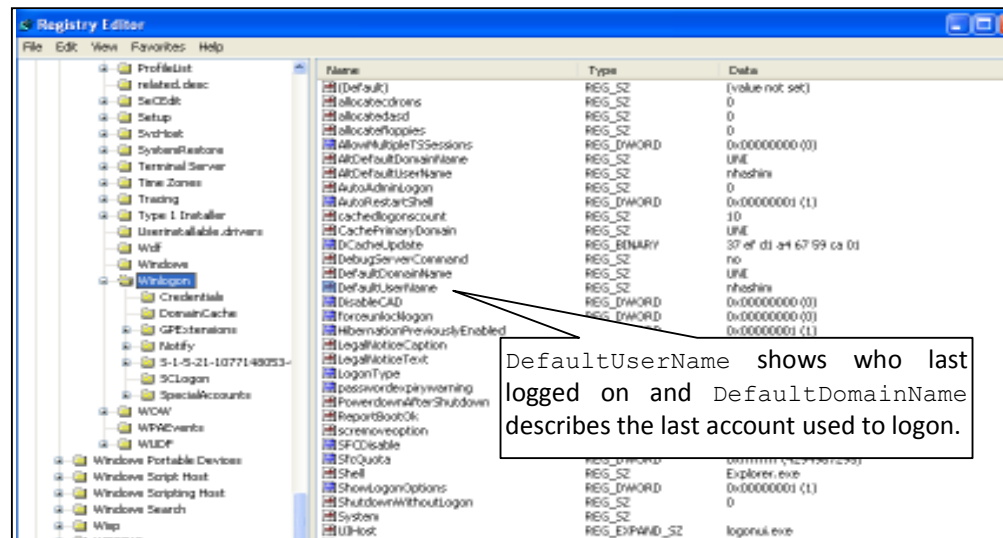


Figure 3.7: The Last-logged-on User and Domain are Stored in the Key HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon

According to Morgan (2009), “*The Windows registry stores a wide variety of information, including core system configurations, user specific configuration, information on installed applications, and user credentials, and is therefore a potentially rich source of evidentiary data*”. This can be seen in the information regarding installed software, last logon and banners. A Registry entry is created and shows software information when programs are installed on a computer, and this information persists in some cases even when the software is removed or uninstalled. The information can be found in HKLM\SOFTWARE. An example of the information held in the Registry relating to users would be the information identifying when the last-logged-on user was doing so using a local account, an account from the local site’s domain, or an account from another trusted domain. The information can be

obtained from HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon. Information regarding the location and contents of banners are useful in investigating a network intrusion. The registry key in which they are stored is HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Policies\System and contains the caption and text of a logon banner that is set on the local system.

### 3.2.3.3 Registry Tools and Related Issues

The interface by which the user primarily views, searches, or modifies the Registry is with the Registry editor tool (regedit.exe), which is installed as part of the Windows operating system. However there are several additional available tools that can be used to examine the Registry: the Advanced Registry Tracer by Elcomsoft (2005) for live Registry, while Guidance Software has EnCase (2005) and AccessData has Registry Viewer (2009) for performing research on compound files of the Registry hives. The built in Windows utility, registry editor (regedit) can be accessed from the run command.

In the registry editor, the left pane is used to view the keys for the registry and the right pane is used to view the values for the registry. It is possible to navigate the hierarchy of the various registry keys and examine the series of values associated with each key. The Windows registry editor has a search function to search for keyword data within the keys, values or data areas of a registry. The Registry editor will not however allow access to the protected areas of the registry where security information is recorded.

The registry is easily damaged (Nelson et al., 2004) and should not be edited or altered via manual editing unless this is essential and the impact of the changes is clearly understood. As the impact of a mistake can render the system inoperable, it is good practice to back up the Registry before making any changes. Furthermore, the registry hive files in an offline mode

are different from the live registry as seen in Windows. The differences are non-volatile keys such as the `HARDWARE` hive file key under `HKLM` (live registry) and the `CurrentControlSet` key for the offline environment (Anson and Bunting, 2007). The operating system displays local times to the user as an offset to Greenwich Mean Time (GMT) based on the user's local time zone offset stored in the Registry. So, an adjustment will need to be made to the timestamps if the time zone of the machine used to examine media is different to the time zone of the examined media (Carvey, 2007).

### 3.2.4 Web Cookie Files As System Generated Artefacts

Web cookies are files that are created by the Web sites visited and sent to the web browser along with the web pages viewed. These web cookies reveal the fact that the user has visited that particular web and may also hold a range of other information.

#### 3.2.4.1 Web Cookie Files Features

The purpose of a cookie may be one or more of the following (this list is not exhaustive):

- To notify the Web site that you have visited it in the past.
- To store a username and password to enable an automatic login when returning to a Web site.
- To record personal user information relating to the website, e.g. name, personal preferences, advertising information, etc.
- To record preferences from previous visits to enable a site to display appropriate adverts.

Web cookies are stored in a specific folder in the hard disk. The location of this folder depends on the version of the Windows and relate to a user account. Table 3.13 shows the typical location of cookie files.

Table 3.13: Web Cookie Files Location

File Name	Windows Version	Location
<username>@<websitename>.txt	Windows 95/98/ME	%SystemRoot%\Cookies %SystemRoot%\Profiles\ <username>\Cookies
	Windows NT/2000/XP	%SystemDrive%\Documents andSettings\<username>\ Cookies\
	Windows Vista	%SystemDrive%\Users\ <username>\AppData\Roaming\ Microsoft\ Windows\Cookies\ %SystemDrive%\Users\ <username>\AppData\Roaming\ Microsoft\ Windows\Cookies\ Low

Cookie files are in human readable ASCII format and in line by line format. The first line contains the variable name and the second line contains the value for the variable name. The third line contains the website name that issued the cookie and the fourth line contains flags. Lines five and six are concatenated and reassemble the expiration time. The next two lines (lines seven and eight) are concatenated and reassemble the creation time. Table 3.14 summarises web cookie files' format.

Table 3.14: Summary of Web Cookie Files Format (Jones, 2003)

Line Number	Fields Description / Summary
1	The variable name
2	The value for the variable
3	The website of the cookie's owner
4	Optional flags
5	The most significant integer for expired time, in FILETIME format
6	The least significant integer for expired time, in FILETIME format
7	The most significant integer for creation time, in FILETIME format
8	The least significant integer for creation time, in FILETIME format
9	The cookie record delimiter (a * character)

### 3.2.4.2 Evidentiary Value of Web Cookie Files

The website usually places information in a cookie in a user's computer to record information about a web session, so it may be requested back at a later date. Cookies are useful as they can be used to automatically sign a user into a specific site and also can hold information about a user's purchases in a web shop. For this reason, cookies aid digital investigators by providing insight into a suspect's Internet activity. For example, after visiting a website such as [www.play.com](http://www.play.com), a cookie will be generated on the user's computer that looks similar as to that illustrated in Figure 3.8.

Cookies can also be used for nefarious purposes since malicious web sites can upload malware masquerading as a cookie to a user's computer (Nolan et al., 2005).



Figure 3.8: Web Cookie File Content

### 3.2.4.3 Web Cookie Files Tools and Related Issues

There are a number of possible tools that can be used to parse and extract information from cookie files. Attempts can be made to secure cookies and to prevent tampering and exploitation by malware, usually by some form of

encryption (MacVittie, 2010). One example of a tool to parse information in a cookie file and return the results in a field delimited format is the Galleta tool developed by Jones (2003). Output from this tool, which is in field delimited format, can be imported to a spreadsheet program so it can be sorted, searched and filtered. This spreadsheet program can also format the data so that it is appropriate for a report.

### 3.2.5 Recycle Bin As A System Generated Artefact

A copy of a deleted file is moved to the Recycle Bin directory when a user deletes a file through Windows Explorer. Although the deletion date and time of a file is not stored in the folder entry, the date and time of deletion is stored in the file. The exact file location again depends on the version of Windows installed on the system.

#### 3.2.5.1 Recycle Bin Features

When a user deletes a file, it is moved to the Recycle Bin. All the original information about the file is stored in a hidden index file, called `INFO/INFO2` that is located in the Recycled folder. The process is: (1) deletion of the file's folder entry in the folder in which the file resides; (2) the creation of a new folder entry for the file in the Recycle Bin; and (3) the addition of information about a file in a hidden system file named `INFO/INFO2`. `INFO` is the file for the Windows 95, 98, and ME while `INFO2` is the file for the Windows NT, 2000, and XP. The Recycle Bin is a hidden system folder and this Recycle Bin directory location is dependent on the version of Windows running on the local machine as shown in Table 3.15.

Table 3.15: Recycle Bin File Location

File Name	Windows Version	Location
INFO	Windows 95/98/ME	%SystemRoot%\Recycled\INFO
INFO2	Windows NT/2000/XP	%SystemRoot%\Recycler\<USER SID>\INFO2
\$R<randomfilename>. <originalfileextension> \$I<randomfilename>. <originalfileextension>	Windows Vista	%SystemRoot%\\$Recycle.Bin\ <USER SID>\

When a user sends a file to the Recycle Bin, Windows records the date and time of deletion and other information, such as the file's location prior to being sent to the Recycle Bin, its index number in the Recycle Bin, and the new filename in the Recycle Bin in the INFO/INFO2 file. Only files deleted by the user and not files deleted by the operating system are in the INFO/INFO2 file. Therefore, the INFO/INFO2 file record indicates that a user knowingly deleted the file. Table 3.16 shows the structure within the INFO2 file.

Table 3.16: The Structure Within the INFO2 File (Jones, 2003)

Offset (bytes )	Size (bytes)	Description
0xC	4	Recycle Record Size
Start of Record+0x04	Variable	Recycle File Name (Null terminated)
Start of Record+0x108	4	Recycle Record Unique ID
Start of Record+0x10C	4	Drive Number for Recycled File
Start of Record+0x110	8	Deletion Date and Time
Start of Record+0x118	4	Deleted Physical File Size

### 3.2.5.2 Evidentiary Value of the Recycle Bin

In order for Windows to restore the file from the recycle bin, certain information must be stored in records so that the original information in the file may be restored. In the case where a digital investigator wishes to determine if a file has been used by an attacker and discarded, the recycle bin is the starting point to recover the file. Examining the recycle bin can determine when a user deleted a particular file, the sequence of deletion and other important file metadata.

The Recycle Bin INFO/INFO2 file records metadata pertaining to a particular file, such as the date of deletion and the original path, and may be useful in confirming or refuting computer users' explanations regarding the presence or history of computer files recovered from their drives.

### 3.2.5.3 Recycle Bin Tools and Related Issues

There are a number of possible tools that can be used to restore the files that have been deleted and stored in the Recycle Bin. In many cases, this data is not entirely over written and by using tools such as ProDiscover (2003); EnCase (2005); FTK (2009); and PC Inspector by Convar Deutschland GMBH ([www.convar.de](http://www.convar.de)), the files can be restored. When a file is deleted and stored in the Recycle Bin, it has not physically moved to the Recycle Bin. The file remains in the original location, but its directory entry has been moved and placed in the hidden folder called Recycled and the deleted file renamed. When a Recycle Bin is used to recover the files, the original path is read from the INFO/INFO2 file, the file renamed and its directory entry restored. This makes recovery from the Recycle Bin possible (Whitehead, 2010).



### **3.2.6 Internet Explorer Activity Files As System Generated Artefacts**

Internet Explorer is a free web browser from Microsoft. A web browser is a software application used to locate and display web pages. The normal data of Internet Explorer browser includes Cookie records, History records and Cache records. These records are saved in files (index.dat) in different directories. Internet Explorer caches websites that a user visits. It stores cached files in the C:\Windows\Temporary Internet file folder. Internet Explorer uses file named index.dat for this purpose in the newer version of Internet Explorer, starting with Internet Explorer 4 and so on. The records in the index.dat file contain the Uniform Resource Locator (URL), the date that the page was last modified by the server, and the date that the URL was last accessed by the user.

#### **3.2.6.1 Internet Explorer Activity Files Features**

The index.dat is a file hidden from computer users that contains every URL and every web page that users have ever visited. The index.dat file in Table 3.17 lists additional areas of the file system where index.dat files can be located. The large scale structure of an index.dat file is the header followed by an array of fixed sized blocks (Chappell, 2010). The index.dat file contains a header that gives important information about the file's properties such as the file length, the HASH table offset and the Internet cache directory names. This is shown in Table 3.18.

Table 3.17: index.dat File Location

File Name	Windows Version	Location
index.dat	Windows 95/98/ME	%SystemRoot%\Windows\ Temporary Internet Files\ Content.IE5\
	Windows NT	%SystemRoot%\Winnt\Profiles\<user name>\Local Settings\Temporary Internet Files\Content.IE5\ %SystemRoot%\Winnt\Profiles\<username>\ Cookies\ %SystemRoot%\Winnt\Profiles\<username>\ Local Settings\History\History.IE5\
	Windows 2000/XP	%SystemRoot%\Document and Settings\ <username>\Local Settings\Temporary Internet Files\ Content.IE5\ %SystemRoot%\Document and Settings\ <username>\ Cookies\ %SystemRoot%\Document and Settings\ <username>\Local Settings\History\History.IE5\
	Windows Vista	%SystemRoot%\Users\<username>\AppData\Local\Microsoft\Windows\History\ History.IE5\ %SystemRoot%\Users\<username>\AppData\Local\Microsoft\Windows\History\Low\ History.IE5\

The HASH table is an array of data that contains entries pointing to the relevant activity data within the index.dat file (Kale, 2007). The HASH table is the “master lookup table” to find valid activity records within the index.dat file. If an index.dat file is large enough, it can have multiple HASH tables which contain a pointer to the next HASH table (Metz, 2009). Each HASH table contains the HASH table length, the pointer to the next HASH table, the activity record flags, and the activity record pointers as in Table 3.19.

Table 3.18: Fields in the index.dat File Header (Thomas, 2003)

Field Name	Offset (bytes )	Size (bytes)	Description
File Length	0x1C	4	This field contains the length of the index.dat file, in 0x80 byte sized records.
HASH Table Offset	0x20	4	This field contains the offset (in bytes) for the beginning of the HASH table.
Cache Directories	0x50	12	This field contains the directories where files are stored that make up the content of the cache. Each directory is 12 bytes long, although only the first 8 bytes are relevant.

Table 3.19: Fields in the HASH Table of index.dat File (Thomas, 2003)

Field Name	Offset from the beginning of the HASH Table	Size (bytes)	Description
HASH Table Length	4	4	This field contains the length of the HASH Table, in 0x80 byte blocks
Next HASH Table	8	4	This field contains the offset (in bytes for the beginning of the file) where the next HASH Table can be located. If the value is zero, this is the last HASH Table in the index.dat file.
Activity Record Flags	$16 + 8*n$ ; where $n = 0,1,2,3...$	4	This 4 byte field contains the flags for the activity record. If the first byte equals 01, the following field does not represent a valid activity record pointer.
Activity Record Pointers	$20 + 8*n$ ; where $n = 0,1,2,3...$	4	This is an activity record offset, in bytes, from the beginning of the file.

The activity records contain the main information to be recovered from the index.dat file. The activity records contain important data fields such as type of activities, length of the activity record, and the data field that is dependent on the type of activity record. REDR, URL and LEAK represent types of activity for the activity records (Mil Incorporated, 2010).

REDR stands for redirection and REDR records are very simple records. The REDR activity record is a statement that the subject's browser was

redirected to another site, there is only simple Internet Web sites data (Thomas, 2003). Table 3.20 shows relevant fields in the REDR activity record.

Table 3.20: Relevant Fields in the REDR Activity Record (Thomas, 2003)

Field Number	Field Name	Offset (in bytes) from the beginning of the HASH Table	Size (bytes)	Description
1	Record Type	0x00	4	This is the field that contains the "REDR" string.
2	Record Length	0x04	4	This is the field that contains the number of 0x80 byte sized blocks that make up the REDR record.
3	URL	0x10	variable	This is the URL, terminated by a NULL (0x00) character.

The URL activity record is a set of data that represents the website a user has visited. A URL activity record contains the record length, the last access and modified times for the activity, the URL offset and data within the record, the filename offset and data within the record, the cache directory number, and the http header offset and data. The LEAK record is generated when an error occurs during the deletion of a URL cache entry (Murr, 2009). The LEAK activity record has exactly the same internal structure as the URL activity record.

In both URL and LEAK types, each record contains an abundance of information, such as record type; record length; last modified time; last access time; the user name; Internet web sites; and cache file path information (Haiping et al., 2009). The fields in the URL and LEAK activity record are summarised in Table 3.21.

Table 3.21: Fields in the URL and LEAK Activity Record (Haiping et al., 2009)

Field Name	Offset (in bytes) from the beginning of the URL or LEAK activity		Size (bytes)	Description
	(version 5)	(version 4)		
Record Type	0x00	0x00	4	This is the field that contains the string "URL" or "LEAK".
Record Length	0x04	0x04	4	This is the number of 0x80 (128) byte blocks that the URL or LEAK record contains.
Last Modified Time Stamp	0x08	0x08	8	This is the last modified time stamp, in FILETIME format.
Last Accessed Time Stamp	0x10	0x10	8	This is the last accessed time stamp, in FILETIME format.
URL or LEAK Offset	0x34	0x38	4	This is the URL or LEAK offset, from the beginning of the record.
Filename Offset	0x3C	0x40	4	This is the filename offset, from the beginning of the record.
Local Cache Directory Index	0x38	0x3C	1	This is the index (starting with zero) of the local directories containing the cache files.
HTTP Header Offset	0x44	0x48	4	This is the offset, from the beginning of the record, where the HTTP Headers are located.

### 3.2.6.2 Evidentiary Value of Internet Explorer Activity Files

One of the potential sources of information that helps to determine the intent of a computer user is to review the Web sites that the individual has visited. Internet Explorer (IE) caches URLs that a user visits in the browser history files and in binary form. When IE is used to browse the web, it keeps a history of its activity that the investigator can use to develop an understanding of the user's activity as well as obtain evidence (Carvey, 2007).

When a user visits any Web site, Internet Explorer first checks to see if it has already stored a local copy of that Web site in the Temporary Internet Folders on the hard drive. If a local copy exists, Internet Explorer uses the

local cached file rather than downloads a fresh copy of the information from the Internet. It also assigns each cached file with an alphanumeric name, and maps the new filenames to the actual filenames in system files. The index.dat file is used to map the cached alphanumeric names to the actual URLs.

### **3.2.6.2 Internet Explorer Activity Files Tools and Related Issues**

The Internet browser history of a live system can be parsed using the WebHistorian tool (Mandiant, 2006) and the ProDiscover tool (2003) can be used for an image. These tools consolidate the browser history information into an easy to view and understand format. Index Dat Spy (Gould, 2004) and Index.dat Analyser (Systemance, 2006) can also be used to view the index.dat contents. Also available is a free viewer, namely Free Internet Window Washer (2010). It works by choosing 'View History' to view the physical index.dat file. A tool to parse information in an index.dat file and return the results in delimited text is called Pasco. It was developed by Jones (2003). Pasco can be run in two different modes: standard methodology and undelete mode. The difference is that the undelete mode ignores the information in the HASH table and reconstructs any valid activity records. The output of this tool will be sent to the standard out (console) by default and can also be imported to a spreadsheet program of choice so it can be sorted, searched and filtered. By importing the output to a spreadsheet program, further analysis of the index.dat is possible and more visualisation techniques can be applied to this data.

## **3.3 Digital Forensic Tools and Techniques**

Digital forensics includes tools as well as techniques that assist in the digital investigation process. Numerous digital forensic tools may be used by digital investigators to search, examine and analyse digital evidence. The

identification and analysis of digital evidence poses unique challenges to digital investigators. A computer system may contain thousands of files any number of which can contain pieces of digital evidence. Each piece of digital evidence may be analysed to identify ownership, location and timing. A number of computer forensic tools are available today that provide a necessary framework for data acquisition and analysis. Britz (2004) categorised these tools into five groups: boot disks, data duplication, verification and preservation tools, data recovery tools; and data analysis tools.

Svensson (2005) used level of operation to group these tools into five categories:

- Bootable Environments Tools: Software that you can use to boot a suspect system into a trusted state.
- Data Acquisition Tools: Tools used to collect data from a suspect's system.
- Media Management Analysis Tools: Tools used to examine the data structures that organise media, such as partition tables and disk labels.
- File System Analysis Tools: Tools used to examine file systems and disk images to recover and view the content of files and folders.
- Application Analysis Tools: Tools used to analyse the file content, for example, viewing log files, images etc.

A tool is used during the acquisition phase to copy data from the suspect hard disk to a trusted device or file. This tool must also preserve the suspect storage device data. In the analysis phase, tools are used to examine acquired data in order to identify pieces of evidence that support or refute hypothesis regarding any incident. In the presentation phase, data from the analysis phase is arranged into a useful format; furthermore, conclusions and corresponding evidence from the investigation are presented. In the forensic process when facing digital investigation tasks, tools are used to look deeper into the data and the analysis is the sum of data applied towards the resolution of the incident (Reyes et al., 2007), therefore these tools among

others should have the functionality of recovering and viewing the content of files.

Table 3.22: Category of Tools by Britz (2004) and Svensson (2005) Studied

	Categories of Tool				
Britz (2004)	Bootable environments tool	Data acquisition tool	Media management analysis tool	File system analysis tool	Application analysis tool
Svensson (2005)	Boot disk Tool	Data duplication tool	Verification and preservation tool	Data recovery tool	Data analysis tool - Indexing - Text searching - Viewers - Time/data verifiers - File managers

Once all the data considered pertinent to the investigation has been preserved, data analysis should be conducted by examining the contents of the files and analysing slack/free space. According to Britz (2004), data analysis tools may be grouped in five general categories: indexing, text searching, viewers, time/data verifiers, and file managers regardless of approach.

Existing tools such as the Microsoft tools supplied as part of the operating system, such as Event Viewer and Registry Editor and others tools explained in Sections 3.2.1 through 3.2.6 that can be used for examination of each of the suggested system generated artefacts. Table 3.22 shows categories of tool by Britz (2004) and Svensson (2005) that are available for analysis of the system generated artefacts of the Windows. This thesis will follow the process suggested by Svensson (2005) and describe tools in the data analysis category.



The next sections provide an examination of commercialised forensic tools and open source forensic tools to provide an insight into what they can offer an investigator. These are specialist tools that are forensically sound that can be used to analyse these generated system artefacts. For example, EnCase and FTK both include facilities to process and examine the Registry. In FTK the Registry Viewer provides the investigator with an expedient and efficient means to view the Windows. The following section reviews tools in term of how they deal with the information stored in Windows system generated artefacts.

### **3.3.1 Commercial Forensic Tools**

A number of commercial vendors provide tools that can be used in computer and electronic investigations and examinations. Among others services included in these tools are: data recovery, indexing/searching and graphical display of results, hashing calculations, and administrative uses. Many commercial products provide online resources that make their use more acceptable in a court of law (Solomon et al., 2005).

#### **3.3.1.1 EnCase**

EnCase available from Guidance Software (2005), provides investigators with tools to authenticate, search and recover computer evidence. That is boot disk tools, data duplication tools, verification and preservation tools, data recovery tools; and data analysis tools. EnCase allows investigators to manage computer evidence, to view relevant files including deleted files, file slack, and unallocated space. This is done by decoding and mounting these files so they are displayed in a logical or hierarchical format (Bunting and Wei, 2006). There are EnCase solutions for acquiring Windows, DOS and Linux.

Guidance Software originally developed EnCase Forensic Edition software, a stand-alone software for acquiring and examining static forensic

images. A further development was EnCase Enterprise, which adopted a server/client model for the business environment. A client side tool would run on the target host so that it could communicate with the forensic examination host over a network connection. This is to avoid changes to the target system if placing the code on the target host and running it (Anson and Bunting, 2007).

EnCase Enterprise Edition also supports volatile data support. This feature takes a snapshot of Random Access Memory (RAM), the Windows Registry, open ports, and running applications. It provides potentially valuable information that is lost when a machine is shut down (Solomon et al., 2005).

According to Bunting and Wei (2006), *“Encase is a fully integrated Windows based computer forensic software application that provides investigators with means of analysing all electronic data contained on computer drives for forensic evidence purposes”*. EnCase allows an investigator to deal with the information stored in Windows artefacts through its following features: read multiple file system format (FAT, NTFS, ext2, ext3, ReiserFS, UFS and JFS); read multiple disk image formats (Raw (dd), VMware, EnCase (.E01) and Safeback); acquire disk images from networked computers; create hash value for files in the case; disk browsing; keyword searching; an integrated viewer allows viewing of many file formats; indexes zip files for analysis of compressed files/folders; and EnScript programming language which automates almost any functionality with complete control over the details.

EnCase uses CookieView as a viewer to examine the cookie folder. CookieView is written by Craig Wilson and when it is installed as a viewer, a right-click on a cookie in EnCase and use the Send To feature will pass the cookie file to CookieView, which will open externally and decode the values in the cookie (Bunting and Wei, 2006).

EnCase can be used to examine the user root folders. However, it does not pull out vital information that the investigator should know, such as what operating system is installed. The folder name that contains the Windows system

files varies with Windows system versions. The investigator still needs prior knowledge of the system artefacts in order to pull out information crucial to an investigation.

Searching features in EnCase are more powerful than those of Sleuth Kit and FTK, but require training to use their full capabilities (Manson et al., 2007). EnCase needs to be customised when using string conditions, EnScript commands, and GREP commands.

### **3.3.1.2 Forensic ToolKit**

The Forensic Toolkit (FTK) by Access Data (2009) is available to the forensic investigator to acquire, decrypt, analyse and report on digital evidence. When an investigator uses the FTK, it allows the system artefacts to be loaded from the image file and this is done through its following features: read file system format (FAT, NTFS, ext2, ext3); read multiple disk image formats (Raw (dd), EnCase (.E01), SMART, Snapback and Safeback); email clients analysis; search feature using keywords; analysis of compressed files/folders by using indexes of zip files; and create hash values for any file in the case.

The FTK has Known File Filter (KFF) feature with multiple different pre-programmed filters for evidence viewing which aids the investigator in focusing on items of interest. For example, an internal viewer allows the investigator to view Microsoft documents (Word, PowerPoint, and Excel); various image files; internal email viewer allows investigator to navigate email from various email storage formats; registry viewing; and password recovery. Viewer for the registry and password recovery come from other company products.

FTK reads various image formats: EnCase, SMART, SnapBack, SafeBack and Linux dd and generates CRC or MD5 hash values (Solomon et

al., 2005). It provides a number of filtering options, including file slack, file system slack, and unallocated space. This can be done through the File Filter Manager of the FTK that enables the user to select which types of files to include (Casey and Larson, 2004). According to Manson et al. (2007), *“the FTK is identified by certain characteristics: substantially less time commitment to training is required to use the program; intuitive GUI design for speedy analysis; lengthy importing process restricts time for analysing of contents of the image; and less customisable compared to EnCase and Sleuth Kit and Autopsy”*.

EnCase and FTK are promoted to withstand scrutiny in a court of law, and hence these are widely acceptable and used by law enforcement and government bodies.

### 3.3.2 Open Source Forensic Tools

This thesis is concerned with developing an open architecture for the forensic analysis of system artefacts. For this reason an appraisal of open source tools will provide a review of the currently available capability of this type of tool. The development of open source tools usually results from an idea to improve on something that already exists or to create something totally new. If the tool is adopted by a significant body of users / developers then it can rival some of the commercial tools. Open source tools provide alternatives to commercial Windows-based forensic tools as they are released under a public licence and can be downloaded from various Internet software development sites (Kavanagh, 2004; POST, 2005; OSI, 2010). One example, ExifTool by Harvey (2004) is free and platform independent software used for reading image metadata, writing and manipulating images. It is both a Perl library and command-line application. Since these types of tool are offered free of charge, this enables them to spread even farther and attract even more developers and the cycle continues (Howlett, 2004).

Open source tools cost very much less compared to commercial tools. The acquisition costs for open source tool is based on time and network throughput spent downloading the software including cost for burning CDs for the acquired software. Therefore, investigators using open source tools can perform their acquisition and analysis at a very low cost. However, the support for open source tools can be limited where public community support for troubleshooting exists but not professional support (Manson et al., 2007). According to Carrier (2002), *“open source tools document the procedures they use by providing the source code, thus allowing the community to accept or reject them. Having access to a tool’s source code will improve the quality of the testing process because bugs can be identified through a code review and by designing tests based on the design and flow of the software”* (Carrier, 2002).

Open source tools are not so easy to learn to use in a short time unless, the user has had previous experience with Linux, and users familiar only with Windows operating system were definitely find them hard to use. Open source allows validation of the source code. Sleuth Kit and Autopsy are both open source tools, promoted as aiding in forensic investigation. Both tools can examine system generated artefacts. Autopsy is able to find web cookies and URLs in the Registry by extracting the SAM file from the Registry, and then Registry Viewer is used to see the web cookies (Manson et al., 2007). According to Manson et al. (2007), *“Open source tools are a very good verification of evidence found using other products and should be included in the academic environment”*.

### 3.3.2.1 Sleuth Kit and Autopsy

The Sleuth Kit and Autopsy Forensic Browser (Autopsy) are both open source tools and run on the Unix and Mac OS X platform (TSK, 2003). Built on The Coroner’s Toolkit (TCT, 1999), The Sleuth Kit contains command line system tools and volume system forensic analysis tools (Solomon et al., 2005). These tools do not rely on the operating system to process the file systems and

therefore deleted and hidden content is shown (Manson et al., 2007). With the volume system tools, partitions are located and extracted and then analysed with the file system analysis tools.

Autopsy is a forensic browser with a graphical interface to the Sleuth Kit. Autopsy is HTML based and shows details about deleted data and the file system structures. Both Sleuth Kit and Autopsy are run in dead analysis and live analysis mode (Solomon et al., 2005). Evidence search techniques used in these two tools are: file listing, file content, hash databases, file type sorting, timeline of file activity, keyword search, metadata analysis, unit data analysis, and image details (Carrier, 2010). However they do have some drawbacks as noted by Kleber and Galvao (2006), *“we identified two negative factors with regard to efficiency: the investigated partition type is limited; and the web interface is not user friendly. These should be considered when adopting them as a main solution for uncovering erased data system audits”*.

Many investigators tend to steer clear from open source software tools, as they are seldom fully supported by their developers, therefore if any problems are faced during an investigation, the investigator may not have the expertise to solve problems posed by the software. However, full commercial licence controlled tools may also face these problems, as often available support is limited.

### **3.4 Problems Facing System Generated Artefacts Analysis**

According to Fry (2007), *“in analysing data, the goals are to highlight its features in order of importance, reveal patterns, and simultaneously show features that exist across multiple dimensions”*. For example, ‘customer’, ‘date’ and ‘product’ are all fields that can be applied meaningfully to a sales receipt. Achieving the goals involves a large quantity of data and the representative nature of the data makes it extremely difficult to gain a comprehensive understanding of its meaning. Each of the system generated artefacts

illustrated and described in the above sections (Sections 3.2.1 through 3.2.6) have these data issues that need to be overcome to be effectively processed and successful forensic investigations.

The collection of tools available to the investigator continues to expand and developers update the tools regularly to enable them to work with the latest technologies. Furthermore, some tools provide a reporting functionality which allows computer forensic examiners to generate reports regarding digital evidence items found, but some tools do not offer a reporting facility. It is the task of the examiner to judge that tools are appropriate for an investigation, and ideally to provide an effective investigation using tools to cross validate the findings rather than highlight differences. Table 3.23 summarises the various key functions present, partially present or, in some cases not present in the tools examined.

So far, the architecture studied in the previous sections does not indicate how data extracted from the system will be interpreted and presented to the investigator. Teerlink and Erbacher (2006); Vlastos and Patel (2007); and Read et al. (2009) have been using visualisation techniques to display information about computer data which can help forensic specialists direct their searches to suspicious files. Such techniques will also prove useful if implemented into forensic tools, allowing the investigator to view data easily, and may result in investigations being performed more time efficiently.

Table 3.23: Comparison of Various Functionalities Provided by the Commercial and Open Source Tools Studied

	Function	EnCase	FTK	SleuthKit and Autopsy
1.	Read only mode	√	√	√
2.	Uses Hash Value for individual file	√	√	√
3.	Easy navigation	√	√	√
4.	Search facility	√	√	√
5.	Includes Hex level viewer	√	√	√
6.	Shows file modified / accessed / created time	√	√	√
7.	Prior knowledge of system artefact required	√	√	√
8.	Locate system artefact files automatically	*	*	
9.	Visual representation of data			
10.	Reporting facility	√	√	
√ - denotes the presence of the functionality * - denotes partial support of some system artefacts, example: Registry				

Table 3.23 extends work by Manson et al. (2007) that shows the main issues regarding the tools available for analysis of the system generated artefacts of the Windows. These tools provide reasonable functionality to the user, but there are a number of areas in need of improvement. Researching the Windows system generated artefacts using the tools available revealed that, in particular, there is a lack of tools for the investigator that can easily extract the information, and present it in a readable form.

Many artefacts are compound in nature. They can be flat files that contain objects (Swap files), they may be flat files that have a hierarchical structure (Registry hive files and Event log files), they may be files that are plain text (web cookie files), or they may be files that are binary (Swap files, Registry hive files and Event log files). This can be further complicated by the format of the data within the files. Unfortunately, the internal structures of



some of the Windows system artefacts are not well known. Only when there is a firm grasp of how data is stored, searching for the data can be performed. Since there is ASCII, binary, hexadecimal encoding and various file formats including extended log file format in specific artefacts, the information contained therein requires deeper analysis.

According to Carrier (2003), *“there are two problem types in analysing acquired data: the issue of data complexity and problem of the quantity of data. The complexity problem arises due to the fact that acquired data is typically in the lowest and most raw format, which is often too difficult for human interpretation. To solve the complexity problem, tools are used to translate data through one or more layers of abstraction until it is understood. The quantity problem arises from the fact that the amount of data to analyse can be very large. Data reduction techniques are used to solve this, by grouping data into one larger event or by removing known data”*.

Through understanding the internal construction of Windows system generated artefacts, investigators will be able to locate them, search the data, present the data, and deliver the information related to them. Investigators will also gain the knowledge necessary to correlate the data among them when they are suspected as avenues of digital evidence, as well as reporting or communicating the results of forensic analysis clearly to decision-makers such as members of a jury, or executives in a company. In short, investigators must have this set of skills if they are going to effectively work with Windows system generated artefacts.

With every new operating system and new version thereof, there are requirements to uncover new system generated artefacts and other such information that proves valuable to the forensic analysis process. For example, in the case of Event logs in the Windows Vista system, besides the three main

categories of event log (Application, Security and System), there are a number of other categories under which different events can be logged. They include Internet Explorer Event Log and Hardware Event Log. Therefore tools and techniques need to be developed that allow forensic analysts to extract relevant and pertinent information from the Windows Event Logs on the Vista system (Carvey, 2007).

The fact that users are not aware of the existence of system generated artefacts leaves the latter open to exploitation. In the case of a system compromise, for example, Microsoft's Internet Information Server (IIS) web server has a number of vulnerabilities that may allow an intruder access. One of the ways to uncover attempts to compromise the IIS web server is to examine the logs generated by the web server. The lack of interaction of the user with these files means these system generated artefacts are an excellent source for an investigator to determine what exactly the user has been doing on the computer.

### **3.5 Conclusion**

In this chapter, artefacts generated by the Windows operating system have been introduced. The complex structures of each artefacts and their value as sources of evidence to the investigator have been highlighted as it is the aim of this thesis to develop a new architecture capable of integrating forensic data from the known and not known internal data structure of the artefacts. The decision has been taken to examine Event logs and Swap files as they are both examples of this type of file. It is therefore proposed that the Event logs (known data structure) and Swap files (not known data structure) be the focus of the thesis and considered for the architecture.

Free forensic tools, open source forensic tools and commercial forensic tools that can be used to examine files related to system generated artefacts have also been described. It has been shown that each tool has its own particular strengths and weaknesses. Some tools provide adequate functionality for the investigative purposes but the investigator needs to have prior knowledge and understanding of the system generated artefacts. The tools cannot support users who lack such knowledge and understanding and hence such ones will be unable to find the data required.

It is this evident that there is potential for the development of a tool that is capable of extracting information from Windows system generated artefacts automatically and providing the results in a manner whereby the investigator does not require prior knowledge of the internal structure of the evidence at hand. Extracting crucial data without the investigators' prior knowledge of such internal structures will also speed up the analysis process and allow investigators to include vital data in their reports. To aid the latter, a reporting facility will be useful to investigators. Chapter 4 explores the proposed architecture designed to address these issues.

## CHAPTER 4

# SYSTEM GENERATED ARTEFACTS FORENSIC ANALYSIS SYSTEM DESIGN

This chapter focuses on a System Generated Artefacts Forensic Analysis (SAFTool) system design. A proposed architecture is introduced which deals with the underlying internal data structure of Windows system generated artefacts. The chapter also discusses the development of that architecture with reference to the hypothesis of the thesis. The architecture requirements and objectives are formalised from issues highlighted in the review of the state-of-the-art tools described in Chapter 3.

### **4.1 Introduction**

In Chapter 3, the number of traces left behind after the usage of the Windows system was explained, highlighting their potential evidentiary value. Chapter

3 also reviewed the current tools available for a digital investigator to process and analyse system generated artefacts. A number of areas where additional functionality could expand the information available to the investigator or increase the efficiency of the investigator were also indicated, the intention being to demonstrate the need for an architecture design that will specifically extract the system related information reviewed in Chapter 3.

Casey (2010) emphasises that investigators not only need to know how to obtain data using forensic tools, but should also be able to analyse the available data for useful characteristics and possible flaws. However, there are concerns over users' knowledge of the location of an artefact, their ability to extract relevant information from the evidentiary sources and make use of presentation techniques or visualisation of data. Although every forensic analysis will have differing aspects based on the dataset, objectives, resources and other factors, the underlying process remains fundamentally the same. Furthermore, the requirements of any digital forensic investigation must also be taken into consideration when designing the tool, so that the tool really is in line with the needs of the investigator in analysing system artefacts.

## **4.2 Existing Structure and Requirements of Digital Forensic Investigations**

Chapter 3 highlighted the general process of a digital forensic investigation; the stages of extracting data and information from computer storage media and the need to assure their accuracy and reliability.

Whether serving customers, clients, prosecutors, or defendants, a computer forensic investigator is required to use the best tools or practices for the given task. According to Bryson and Stevens (2002), the basic requirements for any digital forensic investigation are the collection and preservation of evidence; formulation of leads; focused searches; temporal analysis; and evidence recovery. Through the use of myriad tools or techniques, and the basic requirements for any digital forensic investigation, the core requirements, which are elaborated upon below can be met. The

objective of this thesis is to devise an architecture and tool to identify digital evidence using scientifically derived and proven methods. According to Marcella and Mendenez (2008), “*scientifically derived and proven methods can be used to facilitate or further the reconstruction of events in an investigation*”.

### 4.3 Windows System Generated Artefacts Forensic Analysis System Architecture

The architecture for the project is required to fully support the hypothesis of finding a solution for the development of a flexible, multiplatform and extensible architecture, which may be used for the analysis and visualisation of Windows artefacts to improve the forensic process. The statement of requirements for the architectures based on the issues and a discussion of the failures of the current-state-of-art tools identified in Chapter 3 are as follows:

- The software to be designed is a program that can be used to investigate/examine a system generated artefact. A system artefact holds a collection of entries, each recording an artefact data structure.
- It must be possible to locate the artefact, access the artefact, extract data from the artefact (locate data structure, extract fields of the data structure, decode fields, interpret the fields of the data structure and reconstruct the fields),
- It must be possible to examine new artefacts using the menu option, to check the availability of a data store, to create a data store, to open a data store containing an existing artefact’s data structure, and to close an artefact’s data store.
- It must be possible to visualise the data of an artefact in an intuitive format.
- Reporting facility to extract visual representation of the data.

Key features of the proposed architecture are as follows:

#### **Flexible and Extensible**

The architecture can be defined as flexible and extensible, as it is designed to be used with various Windows system generated artefacts. The data store must also be extensible to accept all sizes of data, and cater for different data types, even without knowing the structure. Someone has to know the structure to develop the database. So the system can be adapted by someone that does know the structure for an investigator who may not know the detailed structured. This includes Windows system generated artefacts that have known internal data structures and not known internal data structures. A modular approach is more appropriate to enable maintenance and improve reliability in use, and to make it easier to upgrade as long as replacement code adheres to the interface specification, then other parts of the code are not affected.

#### **Multiplatform**

Multiplatform promotes the portability of the architecture. This means, the use of a development programming language(s) that allows the system to be run on different hardware with a different operating system.

This architecture concentrates on the examination of the system artefacts' data structures and transforming the data into structured form, thereby, helping the investigator by automating the time-consuming aspect of low-level analysis of the system file format and related data complexity.

The proposed architecture is broken into four key stages, each stage deals with a process of the architecture:

- i. The first stage deals with the plugin for the analyser and visualiser of the artefact to enable investigator to analyse and visualise data of an artefact. This stage of the architecture develops various plugins to analyse and visualise data in different artefacts and represent this data using various techniques. The plugin can be developed independently of the architecture and plugged into the architecture at this stage.

- ii. The second stage will be the analyser to identify and extract data in each of the artefacts being processed. The analyser will need to be easily reconfigurable to interpret the variety of artefacts. Before this stage, the artefact to be processed is prepared. The artefact to be processed comes in one of these formats: exported from imaged hard drive using tools such as Mount Image Pro or FTK, acquired from a Digital Evidence Bag container of digital evidence obtained from disparate sources, or stored using the Advanced Forensics Format (AFF) to indicate imaged disk storage and compressed data used to store digital evidence. In Section 3.4, it was outlined that the current state-of-the-art tools require the user to have a degree of prior knowledge of the complex structure of the artefact they are examining. If the user does not know the details of the complex structure, areas of the structure may be omitted during analysis by mistake, and vital data may not be found. Locating the complex data structures automatically does not require the user to know the exact location of information within the original files. The aim of this stage is to extract all the data from any data structure and parse the raw data ready for insertion into the database. Next, is to invoke the database as a data store that enables data to be stored, queried and retrieved. The database is used because the data needs to be shared, updated, and enable rapid query and retrieval for further analysis and scrutiny. This stage of architecture provides an interface by which the user can insert and extract data to and from the architecture. The database will have read-only privilege, therefore, no data can be changed while being stored in the database. The architecture will incorporate mechanisms to ensure that no data in the original complex structure will change to ensure that forensic integrity is maintained during the investigation. The architecture will hash the files before entering the database, thus allowing checks to ensure the data is not altered. Once the data is processed (stage one), a data store must be available to store the



data. The data store must be available to store data in a manner to enable rapid query and retrieval. The data store must also be extensible to accept all sizes of data, even without knowing the structure. It must be flexible to store all data processed from all data structures (known and not known) with recursion and without recursion. The rapid query and retrieval of data will support the search functionality of the architecture, visualisation of the data, and extraction of the data for reports or visualisation for future analysis. All data retrieved from and inserted to the data store will be in a standardised format to enable tools to operate independently of changes to the database format.

- iii. The third stage in the process is to transform the data into an easily readable format. It is a mechanism whereby data is queried from the database, data representation and further analysis. This is used to structure the data into a narrative construct. It is proposed that the architecture will present the data in the structure visually in many formats. The method in which the data will allow for further analysis and finally present the data in graphs, charts or illustrations. This range of formats will allow the user to easily identify valuable meaningful data in the investigation thus reducing analysis time. This stage deals with visualising the data that is retrieved in an intuitive format to enable investigators to extract evidence. This stage of the architecture represents data using various visualisation techniques and aids such as graphs, charts and keyword searching function.
- iv. The fourth stage, it is proposed that the architecture provides the investigator with a reporting feature. This will allow the investigator to tag various pieces of data; shows visual representations of data; correlate various search results; add

various notes and other data related to the case; and compile all information into meaningful reports.

The architecture was chosen based on the assignment of functionality to stages/components of the architecture as it proved to be the most usable, extensible, and did not complicate any structures further. These stages of the architecture were established by breaking the end result into building steps. Each step of the architecture can be developed independent of the next. Each stage of the architecture must provide functionality to the rest of the architecture. Each stage of the architecture is a step towards to the end result and each were used during the development of the architecture.

The use of a file within a file system for keeping data and other possible database designs are not used as this file is only appropriate for a simple application and for storing data that does not need to be shared and updated by many users; and other database designs tested did not include the use of recursion and resulted in more fields being added to the database records. For example, such records included use of identifiers, keeping note of the parent child relationship found in the artefact complex structure (e.g. Registry).

The top level design of the Windows System Generated Artefacts Forensic Analysis system architecture is shown in Figure 4.1.

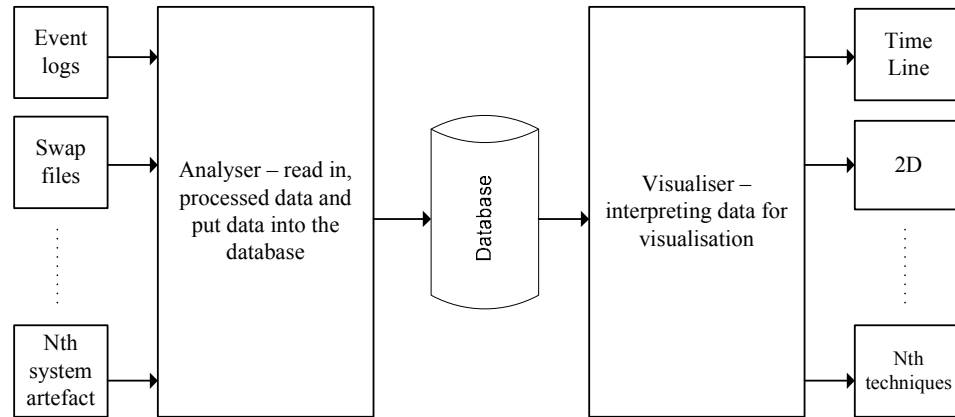


Figure 4.1: High Level Description of Overall Architecture

The key features of the proposed architecture are therefore the analyser, the database, and the visualisation subsystem. This design enables the design and development of a subsystem independently of other subsystems. The analyser component first reads in the evidence item and extracts the data that exists in the original native format in complex structure. This stage will extract all the data from the complex structure. The user should not need prior knowledge of the structure. This performs the automated analysis. Once processed, the data is sent to the database to allow for later visualisation.

Part of the initial system configuration for processing a particular artefact, in addition to providing the analyser with information on the structure of the artefact, will be configuring the database. The database must be available to store data and this data should be able to be rapidly retrieved and queried. It must be extensible in terms of accepting data without any knowledge of the structure and extraction of the data for display or report. This database can be regarded as an interface between the two components to

communicate in a common language that overcomes any specifics related to the syntax and semantics of the data set.

The proposed design of the visualiser module is intended to support the investigator in analysing, interpreting and understanding the data generated from the analyser component and to output the data for the investigator to explore in more detail. This visualiser provides the interface to the database, to display the artefact data in selective views and thus focus on particular aspects of the data or logical flow of interest to the investigator.

### **Design Justification**

The development of a layered architecture for a system is proposed, dividing the software into a number of subsystems. A subsystem typically groups together elements of the system that share some common properties or possibly some common functions (Rational Software Corporation, 2003). An object oriented subsystem encapsulates a coherent set of responsibilities in order to ensure that it has integrity and can be maintained. Furthermore applying the concept of multiple subsystems produces smaller units of development. If these have clearly defined roles and boundaries this can help to maximise reuse at the component level. This also improves maintainability, aids portability between platforms and can assist the software developers in understanding the complexity of large software systems.

The development of multiple subsystems and smaller units of development, enable the design and development of a subsystem independently of other subsystems. It can also enable the extensibility of the final application with additional subsystems being included where an interface has been clearly specified and an overarching design permits the expansion of the application to include new functionality. Dividing a system into subsystems is an effective strategy for handling complexity and splitting a system into subsystems helps to maximise reuse at the component levels, as each subsystem may correspond to a component that is suitable for reuse in

other applications. An effective design in terms of the decomposition of the application into subsystems can reduce the impact on the overall system of a change to its requirements; therefore the use of subsystems can improve maintainability. Moving an application from one implementation platform to another platform can be much easier if the software architecture is appropriate. For example, the conversion of a Windows application so that it can run in a Unix environment requires changes to the software that implements the human computer interface. If this is dealt with by specialised subsystems then the overall software change is localised to these subsystems. As a result, the system as a whole is easier to port to a different operating environment.

It is proposed that the architecture for the Windows System Generated Artefacts Forensic Analysis system have four layers: the presentation layer, the application logic layer, the domain layer and the database layer, as illustrated in Figure 4.2. The presentation layer will be responsible for the differing user interface needs of different artefacts; the application logic layer will be responsible for the control; the domain layer will be responsible for the common functionality, or, to retrieve and store data; and the database layer will be responsible for the data management in order to get data from a database. The approach that has been adopted during the analysis activity of use case realisation results in the identification of boundary, control and entity classes. These boundary classes are mapped onto the presentation layer, the control classes onto the application layer, and the entity classes on the domain layer. Control classes reside on the user machine and manage the interaction between users and the boundary classes and handle the interaction between the business logic of the application and the entity classes and the associated data management classes. A design based on the Broker pattern will be used to handle this (see Section 5.3.4).

The idea of separating out user interface classes from the business and application logic classes and from mechanisms for data storage is to keep the behaviour of the interface separate from the behaviour of the classes that provide the main functionality of the system. The database layer provides access to a database. This four layer architecture separates responsibility for the user interface, the application logic, the domain classes, and the database. Each layer corresponds to one or more subsystems, which may be differentiated from each other by differing levels of abstraction or by a different focus of their functionality. For the Windows System Generated Artefacts Forensic Analysis system, as illustrated in Figure 4.2, the Artefact Database layer provides access to a database that contains all the details of the artefacts, their names and data.

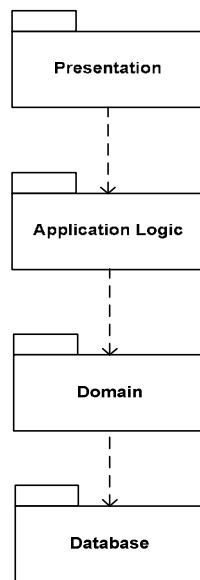


Figure 4.2: Four Layers Architecture Applied to Overall Architecture

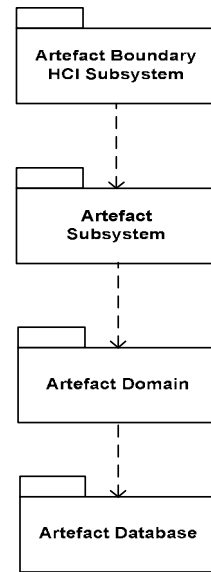


Figure 4.3: Four Layers Architecture Applied to Windows System Artefacts Forensic Analysis System

The Artefact Domain layer uses the Artefact Database layer to retrieve and store data in the database and provides common domain functionality for the layers above, that is the Artefact Details subsystem in the Application Logic layer. The Artefact Details subsystem uses some of the same common domain functionality when detailing an artefact. Each application subsystem has its own presentation layer to cater for the differing interface needs of the different artefact types. In the presentation layer design, one of the aims is to isolate the entity classes in the system from the way that they are presented on screen and in reports and documents. By this means the reusability of the classes can be maximised. With all the design justification mentioned in the above section, the architecture in Figure 4.4 illustrates how the layer architecture relates to the overall architecture in Figure 4.1.

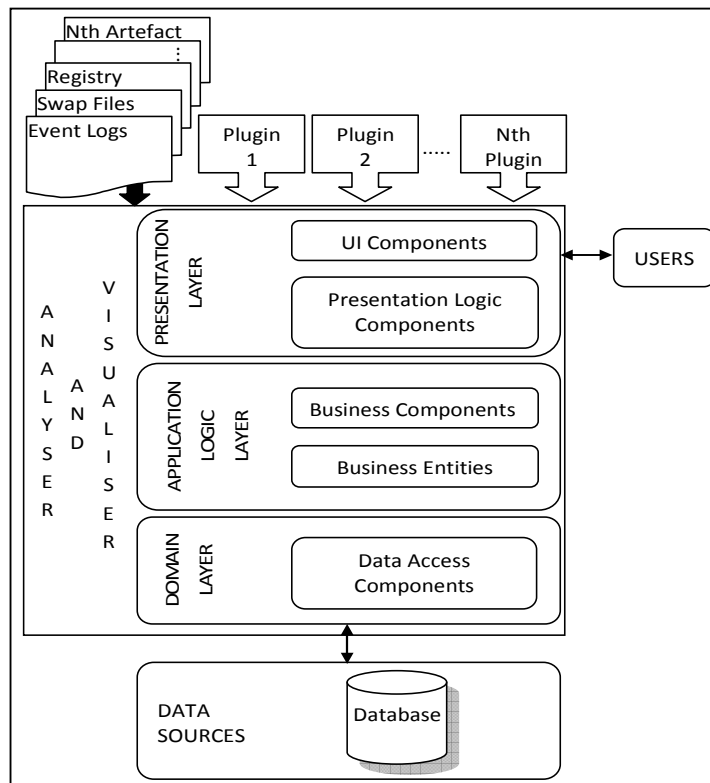


Figure 4.4: Layer Architecture Relates to the Overall Architecture

Figure 4.5 below shows the architecture model based on different views (the package view of the architecture) in order to reason out the proposed system and the way it will operate from different perspectives.

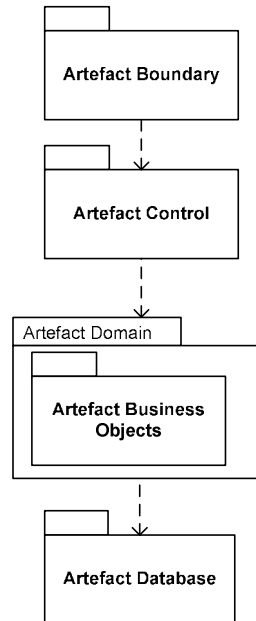


Figure 4.5: Package Architecture Applied to the System

The architecture is required to fully support the hypothesis of finding a solution for the development of a flexible and extensible architecture, open source and prototype tool, which may be used for the analysis and visualisation of Windows system generated artefacts to improve the forensic process. The different layers within the architecture reflect the processing of data that takes place as it is extracted, stored, queried, and visualised.

In order to produce the explicit architecture, the non-functional requirements, the context of the system and how it and its components are used and may be further developed in the future are taken into consideration.



## 4.4 System Analysis

The analysis activities produce the model, and this model shows what parts are in the system and how those parts are related to one another in the system. The analysis model for the Windows System Generated Artefacts Forensic Analysis system is illustrated in Figure 4.6. This model is used to come out with the requirements and specification of what the proposed system will do based on the requirements.

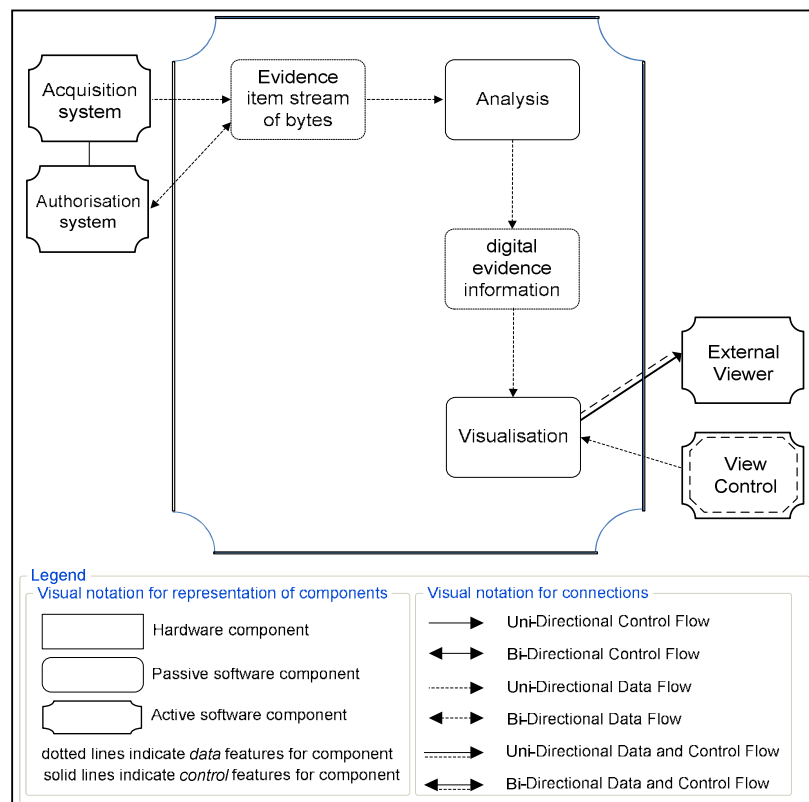


Figure 4.6: The System Analysis Model

## **4.5 Windows System Generated Artefacts Forensic Analysis (SAFTool) System Requirements**

The specific forensic analysis requirements need to be clearly understood when analysing the requirements for the system design. The key requirements highlighted in the previous Section 4.3, is the need for the system to analyse and visualise data of Windows system artefacts. This may require some form of data storage to enable the user to query and then visualise the data in an intuitive format to enable users to extract evidence.

Using the previously stated evidentiary values, features and issues of windows system artefacts tabled in Chapter 3, a list of requirements can be generated. These requirements can be divided into functional requirements, and non functional requirements (security requirements, software quality requirements and other requirements). What a system does or is expected to do is the system functional requirements, while non functional requirements include security requirements, software quality requirements, and other requirements later describe how well these non functional requirements support the functional requirements.

Considering the focus of this project is data analysis, the software development process begins with a statement of requirements, the process then proceeds through analysis, overall design, detailed design and implementation. La Bella (2004) suggested the purpose and value of data analysis was being able to extract different types of data and then turn that data into valuable information. A discussion of the weaknesses of the current state of the art tools used to extract and analyse data was identified in Chapter 3. These weaknesses and other considered issues regarding the artefacts can be used as the requirements of the architecture.

### 4.5.1 Functional Requirements

At this stage, the functional requirements are set out to establish what the system must do. The functional requirements include:

- Descriptions of the processing that the system will be required to carry out.
  - The Windows System Generated Artefacts Forensic Analysis system automates (gets data into the system, controls of the system, and gets data out of the system) the analysis of evidence items (system generated artefacts). This automated system helps the investigator to glean the content information (which is textual data resident in the evidence item) of an evidence item rather than usage information (which indicates how the evidence item or data on it was used) in a forensic analysis investigation.
  - The Windows System Generated Artefacts Forensic Analysis system automates the analysis and visualises the system generated artefacts, thereby relieving the investigator from the time consuming and tedious aspect of low-level analysis. The Windows System Generated Artefacts Forensic Analysis system is designed to facilitate the analysis and visualisation of forensic data in various types of file format and data complexity.
  - The Windows System Generated Artefacts Forensic Analysis system must be able to locate the internal structure of each artefact file for analysis. Locating the internal structure automatically does not require the user to know the exact location of information within the native files.
  - The Windows System Generated Artefacts Forensic Analysis system must facilitate data extraction, data interpretation, and data

reconstruction from the internal structure of the artefact without the user's prior knowledge of the internal structure.

- Additional functionality should be added through plugins or modules, as well as scripting capability via an extensive API.
- Details of the inputs into the system.
  - The main function of the system is to analyse forensic evidence (system generated artefacts) that are extracted from hard disk images.
  - The system should be flexible by being able to analyse evidence items collected from a number of platforms, and from many different sources. The design shall be modular; it should be easy to extend to support new kinds of targets and new types of analysing or processing of data.
- Details of the outputs that are expected from the system.
  - Presentation functionality and reporting functionality are required.
- Details of data that must be held in the system.
  - The Windows System Generated Artefacts Forensic Analysis system must store data in a forensically sound manner whereby no changes can be made to the original data. Forensic integrity is maintained by ensuring no data in the original internal structure of the system generated artefacts will change.
  - No data can be changed while being stored in the system.

In Chapter 3, a discussion of the limitations of the current state-of-the-art tools were identified. These limitations can be used to inform the requirements of the system, thus the system must satisfy these and in doing so, demonstrate an advance on the currently available software tools. Further, the non-functional requirements are also formulated. These non-functional requirements are concerned with the how well the system performs rather than what it does.

### 4.5.2 Security Requirements

Data confidentiality:

The data within the evidence item is considered sensitive in nature, where an attacker can gain access, through accessing the database. Possible methods shall be considered to protect data within the evidence item from being disclosed to unintended parties.

Data integrity:

No data in the original complex structure of the evidential artefact will change.

### 4.5.3 Software Quality Requirements

Verifiability: The system should support the established, standard procedures for handling digital evidence as outlined in Section 2.3 in Chapter 2.

Scalability: The architecture should also be scalable. It should be possible to add visualisation techniques capability to additional evidence items as the need arises. As discussed in Chapter 3, with every new operating system and new version thereof, there are requirements for uncovering new artefacts and other such information that proves valuable to the forensic analysis process.

Usability-1: The system must be easy to use by non-expert users (simplicity).

Usability-2: The system must be easy to use by using keyboard and mouse to control the system.

Accuracy: Accuracy of the generated information output from the analysed files and shown using a textual visualisation

technique is paramount. As this is a key requirement of forensics tools as outlined in the discussion in Section 2.6 in Chapter 2, the analysis phase of the digital forensic phase is the point at which the data is explored in more detail and where the data is drawn together and the analysis is the sum of all data applied towards the resolution of the incident.

Reliability: The system must be reliable, that is no major break on underlying core functionality, which is the resistance to failure of the system.

Extensibility-1: The system must be extensible. It should be easy to add support for new types of data sources for analysis. As this is a key requirement of forensics tools as outlined in the discussion in Section 2.6 in Chapter 2, this is critical since new versions of operating systems and applications are continually developed and the system has to be able to plug in these new sources with as little work as possible.

Extensibility-2: Additional functionality can be added through plugins or modules, as well as scripting capability via an extensive and usable API. As integration with other products, allowing the architecture to be used freely with other product.

#### **4.5.4 Other Requirements**

OR- 1: The system must allow dynamic reconfiguration. This making the plug and play features possible which increased the system flexibility as it is designed to be used with various Windows system generated artefacts.

- OR-2: The architecture and implementation should be independent of underlying software as possible.
- OR-3: Authentication software is used to prove that the evidence has not been changed. Algorithms like MD5 or SHA-1 are required.
- OR-4: There are many document formats and it can become very tedious to try and find the correct program to read the file. So, a document examiner that will read many different file formats should be used.
- OR-5: The tool must be designed in such a way that it provides a form of integrity assurance or record when its utilities are executed, such as it provides an audit record about timestamp or actions taken, or results return from running those utilities. This requirement needs to be addressed, as this is compliant with the best practice requirements for forensics tools as discussed in Chapter 2.

Comparison of the architecture with current state-of-the-art methods shall prove the superiority of the architecture over the available solutions. It is possible to evaluate the outcome of the architecture by:

- Compiling a list of objectives for the architecture and those that will be addressed in the prototype tool.
- Ensure that the architecture satisfies all the objectives outlined. Prove that no such approach is available to date that satisfies the list of objectives to date, therefore, clear proof of contribution of science.
- Perform comparison of the prototype tool against the outlined tools.

## 4.6 Windows System Generated Artefacts Forensic Analysis System Design

According to (Bennett et al., 2006), “*design is about producing a solution that meets the requirements that have been analysed and the design activity is concerned with specifying how the new system will meet the requirements; system design is concerned with the overall architecture of the system and setting of standards; and design is about translating every aspect of the analysis model into a design model that will effectively implement the requirements*”.

According to Zhu (2005), an effective system design is:

**Well structured:** The design should be consistent with chosen design principles. This is to organise the structure of the design.

**Simple:** The design should be as simple as possible.

**Adequate:** The design should meet the stated requirements.

**Efficient:** The functions provided by the design should be computable by using the available resources.

**Flexible:** The design should be able to accommodate likely changes in the requirements if there are requirement changes.

**Practical:** Each module in the design should provide the required facilities, neither more nor less.

**Implementable:** The functions offered by the design should be theoretically computable with the information available and achievable using currently available software and hardware technology.

**Standardised:** The design should be represented using standard or well-defined and familiar notation for any documentation.

The system design should provide a solution to the problems outlined in the previous chapter. The architecture design should give an insight into the high-level design of modular components and how they interact, thus



providing a foundation to build the solution to solve the problems outlined. Further, the design of the system should address the requirements outlined in the previous section of this chapter.

The functional requirements for the system are documented in the use case (see Appendix C). The system is structured as an assembly of interacting objects. This use case captures an element of functionality of the system, whereas requirements that apply to the whole system are captured in a list of requirements. Each use case is a different way to use the system and the completion of each use case produces a different result accordingly.

The initial documenting of the use cases suggests that the following will be part of the system as shown in Figure 4.7 below:

- **Artefact (entity object):** representing the current artefact which the software program is working with.
- **Artefacts (entity object):** representing one of the artefacts that is in the current artefact where there is an arbitrary number of entity objects.
- **ArtefactGUI (boundary object):** representing the interface between the artefact system and the user.
- **ArtefactDbaseSystem (boundary object):** representing the interface between the artefact system and the database system.
- **ArtefactController (controller object):** carries out the use cases that user's choice of action.

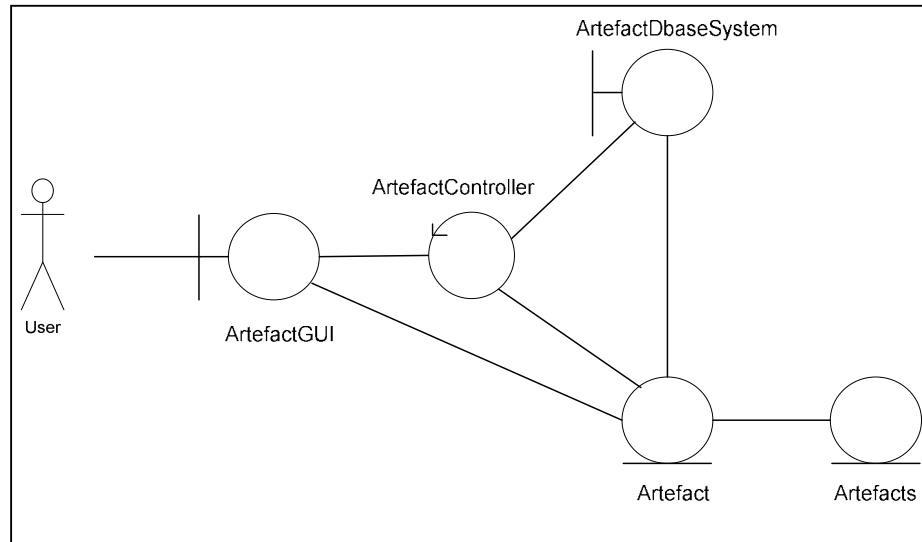


Figure 4.7: Collaboration for the Windows System Generated Artefacts Forensic Analysis System

The various use cases work with these objects, as follows (see Appendix C):

- The *Analyse New Artefact* Use Case involves creating a new *Artefact* object, obtaining the new information, and then instructing the *Artefact* object to add the new artefact with this information to its collection.
- The *Visualise Artefact Data* Use Case involves getting a database specification (all data retrieved from and inserted to the data store will be in a standardised format to enable tools to operate independently of changes to the database format) from the user, then telling the *ArtefactDbaseSystem* object to read in an *Artefact* object from this database, and then display the current information about the desired artefact.

In the model-view-controller design pattern, responsibilities are assigned to the various classes. The two entity classes, *Artefact* and *Artefacts* serve as the model. The GUI class, which is the *ArtefactGUI*, serves as the view. The controller class, *ArtefactController*, serves as the controller.

The view, ArtefactGUI, needs to be made an observer of the model, Artefact, so that it always reflects the current state of the model.

Class Responsibility Collaboration (CRC) cards, as used by Bennett et al., (2006) and shown in Table 4.1 are used to assign responsibilities to various classes and the collaborations that are necessary to fulfill the responsibilities for the tasks required by the various use cases.

Table 4.1: Class Responsibility Collaboration for the Main Classes Contained in Windows System Generated Artefacts Forensic Analysis System

Class	Responsibility	Collaborators
Artefact	Representing the current artefact which is the software program is working with.	-
ArtefactController	Carries out the use cases that user's choice of action.	-
	Perform the Analyse New object is to carry out the use cases that user's choice of action.	Artefact ArtefactDbaseSystem
	Perform the Visualise Artefact Data use case.	Artefact ArtefactDbaseSystem
ArtefactGUI	Allow interaction between the program and the user.	-
	Keep track of the artefact object it is displaying.	-
	Displaying a list of the artefacts in the current artefact.	Artefact
	Displaying the title of the current artefact. – if any	Artefact
ArtefactDbaseSystem	To manage interaction between the program and the database system.	-
	Read a stored artefact from a database, given its database name.	Artefact
	Save an artefact to a database, given its database name.	Artefact
Artefacts	To maintain information about a single artefact.	-
	Create a new object.	-

The classes discovered during system analysis and additional classes discovered during design, suggest that the following will be the classes required in the class diagram for the Windows System Generated Artefacts Forensic Analysis system:

- *ArtefactApplication* – responsible for creating the *ArtefactDbaseSystem* and GUI objects. It is the main class for the application and starts up the application.
- *ArtefactListener* (a utility class) – responsible for the user's choice of action.
- *AnalyseNewArtefact* – action for analysing artefact objects (used for recovering data structure).
- *VisualiseArtefactData* – action for visualising artefact objects (used for visualising data)

Figure 4.8 is the class diagram for the system where only the name of each class is shown, the attribute and behaviour compartments are omitted. The relationships hold between the objects is follows:

- *ArtefactApplication* object is responsible for creating *ArtefactController* object.
- *ArtefactDbaseSystem* responsible for saving and reloading an artefact.
- *ArtefactController* object responsible for creating *ArtefactGUI* object and this *ArtefactGUI* object is responsible for keeping track of its current artefact object displayed.

- Artefacts object is responsible for creating and keeping track of record objects, where there is an arbitrary number of entity objects.
- ArtefactListener object is used by the ArtefactController to allow the user to choose which action to perform in examining an artefact.
- Artefact object use the appropriate action object when action is chosen.

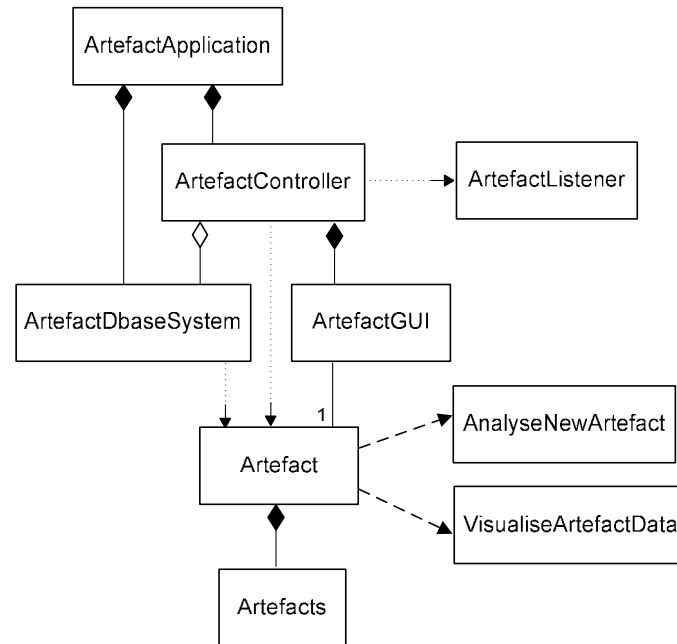


Figure 4.8: Objects for the Windows System Generated Artefacts Forensic Analysis System

## 4.7 Operation and Control Specifications

Operation specifications are detailed specifications of the behaviour of a system model. An operation specification is a framework for a more detailed design specification, which can then be used to derive an appropriate

implementation of the operation in code. It can also be used to verify that the method does indeed meet the original specification, which in turn describes what the users intended, thus checking that the requirements have been implemented correctly.

An operation that translates a stream of bytes into a usable file structure of an artefact and recovers the content of the file is invoked during the use case `Analyse New Artefact` (in Appendix C). For ease of reference, the use case description is repeated below:

*The artefact's data may be recovered from a stream of bytes of the internal structure of an artefact to perform artefact analysis.*

A sequence of steps for this operation will include the following steps for recovering data structure.

1. Locate: Locate the fields (the units of information) within the structure. Different method of locating fields for different files of the Windows system generated artefacts. Examples of the methods used are:
  - i. Fixed offset
  - ii. Calculation
  - iii. Iteration
  - iv. Location by outside knowledge from some source
2. Extract: To extract the data of the internal structure out of the stream of bytes.
3. Decode: After the relevant data has been extracted from the internal structure, it is still possible that further extraction may be necessary, specifically the bit fields. It will use a list of bit fields to check on each field in the stream of bytes. Examples of bit fields are: flags, attributes, date field, time field, etc.

4. Interpretation: Takes the output of the previous step or the extract step and performs various computations. Examples: the value for the years of date field and second of time field need to be interpreted.
5. Reconstruction: Information from the previous step (interpretation) is used to reconstruct a usable representation of the internal structure or fields.

Figure 4.9 shows the activity diagram for the use case *Analyse New Artefact* and illustrates the internal operation logic of the process.

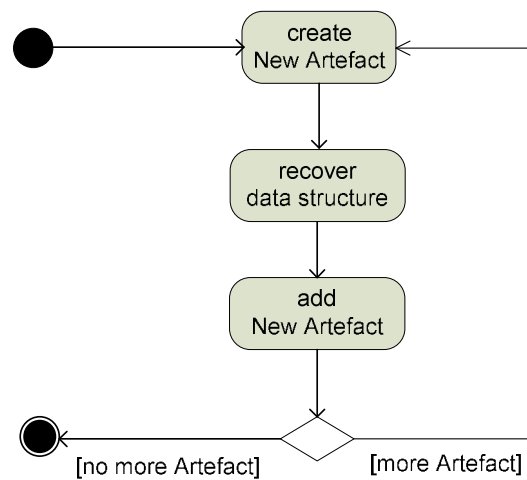


Figure 4.9: Activity Diagram for Use Case *Analyse New Artefact*

The operational process that maps data to a visual form, that is, the data representation is invoked during the use case *Visualise Artefact Data* (in Appendix C). For ease of reference, the use case description is repeated below:

*The artefact's data may be examined to identify valuable meaningful data and allow for further analysis. The mapping of data to a visual form presents the data in a more human friendly format by the use of graphs, charts or illustrations.*

A sequence of steps for this operation will include the following steps for understanding data.

1. Acquire: Obtain the data, whether from a file on a disk or a file over a network; or database.
2. Parse: Give some structure for the data's meaning, that is, to change it into a format that tags each part of the data with its intended use. Each record of the file must be broken into fields.
3. Filter: Remove any extraneous data.
4. Mine: Add methods as a way to shows patterns.
5. Represent: Determines the basic form of a visual model that a set of data will take.  
Example: Bar graph, tree, table and so forth.
6. Refine: Design methods are used to improve the basic representation to make the information presented clearer and more visually useful.
7. Interact: Apply methods for manipulating or controlling the data.
8. Output: The information file is translated into the required output format and output to the screen.

Figure 4.10 shows the activity diagram for the use case `Visualise Artefact Data` and illustrates the internal operation logic of the process.



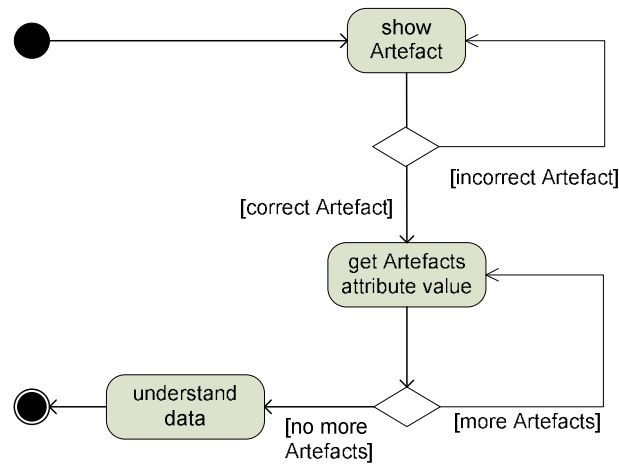


Figure 4.10: Activity Diagram for Use Case Visualise Artefact Data

The activity involved in the use case realisation of the system is next presented using the following UML diagrams: communication diagram and sequence diagrams that reflect understanding of the domain and of the requirements. The communication diagrams and sequence diagrams illustrating the design for the entire prototype tool can be found in Appendix D. The sequence diagrams show the interaction between users, systems and sub-systems; with the emphasis on the ordering of time of messages. The communication diagram shows similar information as the sequence diagram, but emphasises the structural organisation of the objects' send and receive messages; thus illustrates the dynamic view of the system.

## 4.8 Data Management

The application needs to store data between one execution of the program and the next. Data must be contained within a form of shared data storage system so that users can retrieve it when access is required. Requirements include the

storage of persistent data to operate independently of any changes made to the database, thus reinforcing the architecture's extensibility.

The aim of using a Database Management System (DBMS) is the DBMS organises and manages the data and decouples the storage mechanisms from the application programs. It is proposed that the object oriented system, that is, the Windows System Generated Artefacts Forensic Analysis System to be built, use a relational DBMS to provide the storage for the system. The classes are converted to tables, the classes are flattened into tables in order to design the storage structures. In this example, the a `ProjectMember` with `EmpID`, `FirstName`, `LastName`, and `Rate` properties; and a `ProjectManager` with `SignatureLevel` and `BudgetAuthority` properties classes are flattened into a `ProjectMembers` table, which groups all properties defined in the class hierarchy table and contains an additional a `EmpType` column. Therefore, now `ProjectMembers` table has `EmpID`, `FirstName`, `LastName`, `Rate`, `SignatureLevel`, `BudgetAuthority`, and `EmpType` (Hautefeuille, 2011). When the system requires an instance of a class from the database, it will have to retrieve the data from all the tables that hold parts of that object instance and reconstruct it into an object. When a complex object instance has to be stored, it will have to be taken apart and parts of it will be stored in different tables. Data modelling and the normalisation approach will be used to flatten out the objects in order to store them in a relational database.

Separate classes are introduced into the system, whose role is to deal with the storage and retrieval of the other classes. These classes are part of the data storage layer. The data storage classes are decoupled from the business classes. The business classes which contain metadata indicate how they are to be stored. The same business classes can be reused unchanged with different storage mechanisms. The classes that provide the data storage services are held in a separate package. The objective here is to separate the data storage mechanisms completely from the business classes. One of the

business requirements is to produce a report combining data from different applications, so there is a need to access the data in new ways.

The database design will be capable of storing all an artefact's file data from the Windows system. A composite of data will be placed in here based on definitions and methods in which the data is to be extracted, and will be specific to each of the artefacts.

Once the data is processed by the analyser, a database must be available to store data so that this data can be rapidly retrieved and queried. The design of this database must be extensible in terms of accepting all sizes of data and extraction of the data for display or printout in the report without knowing the structure. This database can be regarded as an interface between the two components to communicate in a common language that overcomes any specifics related to the syntax and semantics of the data set.

## **4.9 Object Oriented Approach to System Development**

The purpose of system design is to answer the question: (Stair and Reynolds, 2008) *"How will the system solve the problem?"*. The primary result of the system design phase is a technical design that details system outputs, inputs, and user interfaces; and shows how these components are related (Arnott, 2006). There are several approaches that can be followed today which have their own justification for users to adopt them, but all attempt to deliver a better system. Such approaches include: structured system analysis and design approach, the prototyping approach, the rapid application development approach, the extreme programming approach, the joint application development approach, the end user system development approach and the object oriented systems development.

One of the important design considerations that can have significant consequences in all areas of the architecture is the choice of the system development approach. The reason for choosing a specific approach is to avoid many of the problems and pitfalls in system development.

By adopting the object oriented approach for implementation of the architecture, the built-in beneficial features and advantages that come with this approach can be adopted and realised. These are the modularity advantages of the system on the ease of maintenance and upgrading, with software broken into smaller pieces instead of a monolithic block of code to deal with all the processes in analysing system generated artefacts; and the reusable advantages of the system is realised on the integration with other products, providing a standard and well documented process for other products for their integration. This is further supported by the fact that this approach combines the logic of the system development life cycle with the capabilities of object oriented modelling and programming.

The beneficial features and advantages of the object-oriented approach are discussed below (Bennett et al., 2006):

#### **Increasing abstraction**

Abstraction in object orientation means (Martin, 2001), *“The elimination of the irrelevant and amplification of the essential”*. The increase in abstraction applies both to the activity of programming and the tasks that computer programs are expected to perform.

#### **Event-driven programming**

The applications of object-oriented programming include event-driven systems such as simulation and graphical user interfaces (GUIs). GUI programs typically are created using event-driven systems and present the user with possible activities. This might include: clicking on a button at the top of the screen, pulling down a menu, and moving the mouse across an icon bar. The object-oriented approach encapsulates data and processing methods together, which is useful for this project in that the system can be informed that an event has occurred, and then manage this by asking a processing method to act on the relevant piece of data. The intended prototype system will have the objective of event-driven system that have the dropdown menu populated with

the names of all artefacts, so the users are allowed to select an artefact to examine from the dropdown menu on the interface.

For example, if a simulation task is to be programmed in a procedural language, it is very difficult to program it effectively in a procedural language since program designs for procedural languages are based on the assumption that the program structure controls the flow of execution. Beside, for a program with functions to tackle the simulation task, it must have separate routines that test for; and respond to a number of alternative conditions. Therefore, object oriented techniques are most effective in event-driven systems such as simulation and GUIs. Before event driven programming, the event handler is implemented as subroutines and in event driven programming the event handler is implemented as processing methods of objects.

### **Graphical User Interface**

Object oriented technology can be employed to anticipate every possible route that a user might take through a system's interface. The number of possible options in a GUI can mean that the majority of desktop applications are now very difficult to design or control in a procedural way. The object-oriented paradigm offers a method to design software, each component of which offers clear services that can be used by other parts of the system quite independently of the sequence of tasks or the flow of control. The intended prototype system will present results using a GUI because the goal of GUI design is to make user's interaction as simple an efficient as possible in terms of navigation, analysis and retrieval.

### **Modular Software**

In an object-oriented system, classes have two kinds of definition: from an external and internal perspective. From an external, a class is defined in terms of its interface, other objects need only know the services that are offered by objects of that class and the signature used to request each service. From an internal, a class is defined in terms of what it knows and what it can do. Modularity means the implementation of each part of the constructed software

is independent of the implementation of other parts of the constructed software. This contributes to solving some of the most intractable problems in information systems development. Advantages of modular software are:

- Maintaining a system built in a modular way is easier, as changes to a sub-system are much less likely to affect the rest of the system.
- It is easier to upgrade a modular system. As long as replacement modules adhere to the interface specifications of their predecessors, other parts of the system are not affected.
- It is easier to build a system that is reliable in use. This is because subsystems can be tested more thoroughly, and fewer problems to be addressed later when the whole system is assembled.
- Implementation of a modular system can be in small, manageable increments. Provided each module is designed to provide a useful and coherent package of functionality, they can be introduced one at a time.

### **Life Cycle**

Object orientation addresses system design by a cyclic development approach. The iterative process of this approach can repeat and this aspect is linked to the modular character of an object-oriented system and also the seamless development of models throughout an object oriented life cycle.

### **Model Transitions**

A successful design is one that meets the requirements in a way that is functional, efficient, and economic. In structured approaches, the designs for new systems are hard to trace back to the original requirements.

Object oriented analysis and design avoid these transition problems by using a set of models throughout analysis and design, adding more detail at each stage. In Unified Modelling Language (UML) (Shnitman, 2000), a modeling language used to support the object oriented approach, the fundamental analysis models are the use case and the class diagram and these continue as the backbone of the design, with other design models derived directly or indirectly from them.

### Reusable Software

Information systems are costly in the case where they are used to invent new solutions to old problems. This has led to the demand for reusable software components. Reusable software components can eliminate the need to keep inventing new solutions. Object oriented development methods offer developers software components that are reusable in other systems. This is achieved through the highly modular nature of object oriented software and also the way that object oriented models are organised. The intended prototype system will have the objective of developing reusable components that are reusable in other systems.

The techniques and notation system chosen to support the object oriented approach is the Unified Modelling Language (UML) (IBM, 2011a). UML is chosen as it has become the industry standard for modelling information systems and consists mainly of a graphical language to represent the concepts that are required in the development of an object-oriented information system. In UML 2.0 (Object Management Group, 2005), a model is defined as: *“A model captures a view of a physical system. It is an abstraction of a physical system, with certain purpose. This purpose determines what is to be included in the model and what is irrelevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the relevant level of detail”*.

The thesis had the objective of developing a reusable architecture, as such object oriented development methods offer developing software components that are reusable in other systems. This is achieved through the highly modular nature of object oriented software and also of the way that object oriented models are organised.

In order to assess the architecture and the system, a prototype tool was developed based on the architecture. Chapter 5 will discuss the implementation of the architecture in relation to the prototype in detail. Chapter 6 will discuss the evaluation and analysis of the architecture and the in relation to the prototype tool and state-of-the-art tools available.

## 4.10 Conclusion

The contribution of this project lies not only in the ability to manipulate and visualise the event logs and swap files, but also in the development of a flexible and extensible architecture to process various Windows generated artefacts. This is based on a clear understanding of the priority of the requirements that have been identified from the highlighted evidentiary values, features of the artefacts, and issues associated with current state-of-the-art tools explored in Chapters 1 and 2.

This chapter has described the design of a software architecture for the forensic analysis of Windows system generated artefacts. Forensic analysis has a number of unique requirements that directly impact on the design of an architecture. For example, the need to interact with multiple disparate data sources led to the development of the plugins. The system architecture is flexible in the sense that other types of artefact (e.g. Registry, Internet Explorer Activity Files, Web Cookie files) could be easily added into the system in addition to Event logs and Swap files. Furthermore, different plugins may be used for processing each of the artefacts. The following chapter details the implementation of the architecture.

An object oriented approach, is elaborated in activities to design the system architecture, identify objects in the system, describe the design using different objects' models, and document the objects' interfaces.



## CHAPTER 5

# SYSTEM GENERATED ARTEFACTS FORENSIC ANALYSIS SYSTEM IMPLEMENTATION

In Chapter 4, the proposed architecture aimed at processing Windows system generated artefacts was introduced and the rationale for the design decision was discussed. This chapter outlines the development of that architecture in terms of a prototype implementation. The architecture requirements and objectives are formalised from the issues identified in the state-of-the-art tools review in Chapter 3. This chapter then highlights some of the key issues in creating a prototype implementation of that architecture.

### **5.1 Introduction**

The initial section of Chapter 4 highlighted the requirements of the architectural design. In terms of the implementation of the prototype system the architecture necessitates the creation or integration of a number of components, including a database management system. The implementation of a prototype system requires a range of tools and languages and vitally a set of documentation to ensure the system can be understood and

extended by other programmers / users. UML provides a number of diagrams, two of which can be used to document the implementation of the system. Component diagrams are used to document dependencies between the different elements of the system. These component diagrams can then be combined with deployment diagrams to show how the software components relate to the physical architecture of the system.

## **5.2 Software Development Tools**

A software development tool is a program or application that is employed in the development, repair, or enhancement of other programs or of hardware (Daintith, 2004). The software development tools used for the implementation of the prototype system include the use of Integrated Development Environment (IDE) that is the NetBeans IDE; object oriented programming language, that is Java; and JavaDB that comes with database management system and connectivity (Java Database Connectivity (JDBC)), that is DBMS. NetBeans IDE is a platform that provides reliable and flexible application architecture. It provides a means to develop applications by the use of components provided with NetBeans IDE itself. NetBeans IDE allows the development of interfaces by allowing the developer to use the generic framework without the need to manually code the interface. Java is used as the programming language as Java is object-oriented. Java programming is object-oriented as is centered on creating objects, manipulating objects, and making objects work together. This allows the creation of modular programs and reusable code. JavaDB was the database used for the implementation of the prototype. This decision was based on the fact that it is an open source Java technology database. It support standards based SQL, and JDBC API that supports creating and executing SQL statements.

These tools are used to support the implementation of the whole system and are discussed below in Section 5.2.1 through 5.2.3.

### 5.2.1 Integrated Development Environment

An Integrated Development Environment (IDE) is a set of tools that aids application development (Nourie, 2005), essentially it is a software tool that can be used to keep track of all files containing source code, resource files and the dependencies between them, recompiling all those that have changed as a project is being built. Therefore, as the IDE combines the features of many tools into single development package, and as discussed in Section 5.2, it is used in the architecture implementation.

Sun Microsystems supports three IDEs for the Java platform: NetBeans, Sun Java Studio Creator, and Sun Java Enterprise. IDEs are important as they provide comprehensive facilities to developers for software development for the implementation of the prototype. It consists of a source code editor, a compiler, an interpreter, build automation tools and a debugger. The IDEs for developing HTML applications are HomeSite (Adobe Systems Incorporated, 2011), DreamWeaver (Adobe Systems Incorporated, 2011) and FrontPage (Microsoft Corporation, 2011). The IDE selected for the implementation of the proposed system is NetBeans. The advantages of using NetBeans are: the NetBeans IDE is open source, written in the Java programming language, and supported by Sun Microsystems. NetBeans, IDE provides the services for creating desktop applications, such as window and menu management features, setting storage and fully supporting Java Development Kit (JDK). NetBeans IDE incorporates a multi-window editor which is the mechanism for managing the files that make up a project, links to the compiler so that code can be compiled from within the IDE, and a debugger to help step through the code to find errors. NetBeans saves time by managing window, settings, and data. This is due to the hypothesis of this thesis was to develop an open source solution and a lot support from the Java community is there if a problem is encountered during the programming phase.

There are several popular IDEs as alternatives to NetBeans IDE, including Borland JBuilder, IntelliJ IDEA, and Eclipse. These were not used for the development of the project because they are not as richly featured as NetBeans IDE. Moreover, NetBeans has a closer relationship with the programs it is developing than most other IDEs due to its managing window, settings, and data (Boudreau et al., 2003). In addition, NetBeans IDE contains drag and drop features that enable the rapid development and revision of graphical user interface components. Finally, to keep the architecture as maintainable as possible NetBeans IDE was the choice due to it incorporates a mechanism for managing the files that make up a project, links to the compiler that compiled the code from within the IDE, and a debugger to help through the code to find errors.

Developing new toolkits requires modular applications. NetBeans IDE provides this feature by having facilities to build multiple modules and create dependency (communicate among themselves) between them effortlessly. A module suite can be created inside NetBeans IDE to create modular applications (Sun Microsystems, 2007).

### **5.2.2 Object Oriented Programming**

An object oriented approach to the problem was considered the most appropriate method as object oriented programming model the physical world, in a way that the building of a computer applications represent how objects are assembled, that is to say, objects are made up of many kinds of smaller objects. By using this object oriented approach of development, the resulting software is understandable, reusable, and reliable (KayKeys, 2005).

Object oriented programming focuses on the task for which the users are using the computers rather than the way a computer handles tasks. It is a software development methodology in which a program is conceptualised as a

group of objects that work together. Classes contain data and method. A method is how to use that data. These data and methods are used when creating objects.

There is a number of possible alternative languages that could be employee to develop the proposed prototype system. The choice of an object oriented system suggests that Java, C++, C#, Python and Visual Basic are possible candidates. All these programming languages provide complex functionality and libraries to develop programs. However to keep the architecture as platform neutral as possible, Java was the language of choice, because this would enable the program to run on different platform without modification.

According to Sun Developer Network (2010), *“Java programs are compiled into a format called bytecode that is run by any operating system, software, or device with a Java interpreter. A Java program can be created on a Windows Vista machine that runs on a Linux web server, Apple Mac using OS X, and Palm personal digital assistant. As long as a platform has a Java interpreter, it can run the bytecode”*.

### 5.2.3 Database Management System (DBMS) and Connectivity

The Database Management System is used to organise and manage the tasks associated with storing and providing effective access to the data. This will enable a degree of flexibility in accessing the data. Tools and features of the DBMS can be used to manage the data: **Data Definition Language (DDL)** is used to specify the data held in the DBMS and the structures that are used to hold it; **Data Manipulation Language (DML)** is used to specify updates and retrievals of the data; **Security** is used to control the access to the data and permissions granted to different users for different levels of access; and **Integrity** is used to specify constraints to ensure that the integrity of the data is maintained.

Structured query language (SQL) is used to provide both the DDL and DML for the relational databases and the main advantage of using this language is that it forms the basis for the user's interaction with relational database systems. Once a user is familiar with the construction of SQL statements they can apply this knowledge to any database supporting the language. SQL is the industry-standard approach to accessing relational databases (IBM, 2006).

The required interaction between the developed tools and the database is provided by the Java Database Connectivity (JDBC). JDBC is a set of classes, and when working with a database in developing an application, JDBC can be used. JDBC classes were developed by a number of organisations including Microsoft (Microsoft Corporation, 2011a), Sybase (Sybase, 2011), Oracle (Oracle, 2011), and Informix (IBM, 2011b). According to Cadenhead and Lemay (2007), *"The JDBC library includes classes for each of the tasks commonly associated with database usage these include making a connection to a database, creating a statement using SQL, executing SQL query in the database, and enabling the viewing of the resulting records"*.

Database management systems provide various facilities that are useful in many applications. According to Bennett et al. (2006), *"a DBMS typically offers support for: different views of the data by different users; control of multi-user access; distribution of the data over different platforms; security; enforcement of integrity constraints; access to data by various applications; data recovery; portability across platforms; data access via query languages; and query optimization"*. The hypothesis of the thesis was to develop an architecture that is able to integrate forensic data from known and not known internal data structures of system generated artefacts by the Windows operating system and to design and implement a proof of concept prototype tool, with appropriate example artefacts. Considering DBMS appear to meet the main criteria of the architecture.

### 5.3 Implementation of the Architecture

According to Stair and Reynolds (2008), “*Systems implementation involves creating or acquiring the various system components (hardware, software, databases, etc) defined in the design step, assembling them, and putting the new system into operation*”.

In Chapter 3 it was determined that the Event logs (a known data structure) and Swap files (a not known data structure) were the chosen Windows system generated artefacts selected for the prototype implementation, as they illustrate the types of complex internal structure found in Windows system generated artefacts. The following section discusses the implementation for each package of the architecture will be discussed, including the development of a class diagram and supporting series of interactions for the package illustrating how the data is extracted and processed throughout the architecture.

The packages have been named in a way that allows the use of Java package notation for classes. For example, the boundary classes will be in the package `SAFTool::Boundary`, the control classes will be in the package `SAFTool::Control`, and the entity classes will be in the package `SAFTool::Entity`. Figure 5.1 shows the package diagram view for the software architecture of the Windows System Generated Artefacts Forensic Analysis system.

The architecture can also be modelled using the structural view model and data view model. A data view is the same diagram as a structural view, but the architecture defines the type of data that is to be provided by one subsystem to another subsystem. Figure 5.2 shows structural and data view of the system under development. The data dictionary presented in Table 5.1 explains the meaning of the entities shown in Figure 5.2.

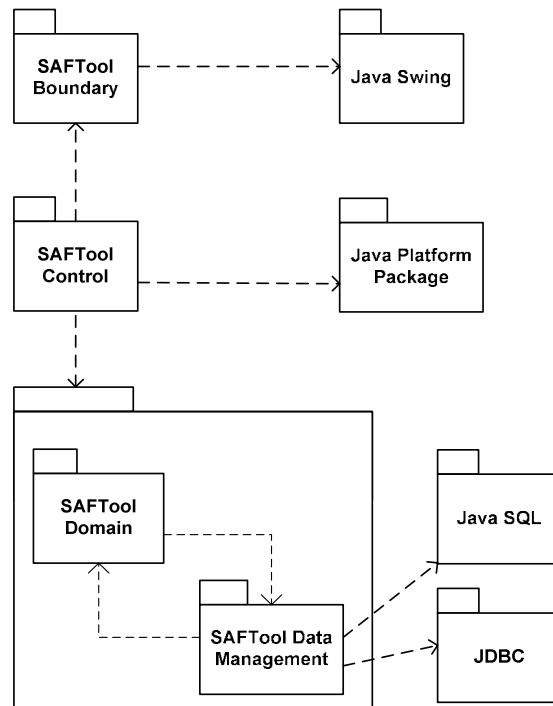


Figure 5.1: Package Diagram View for the Software Architecture



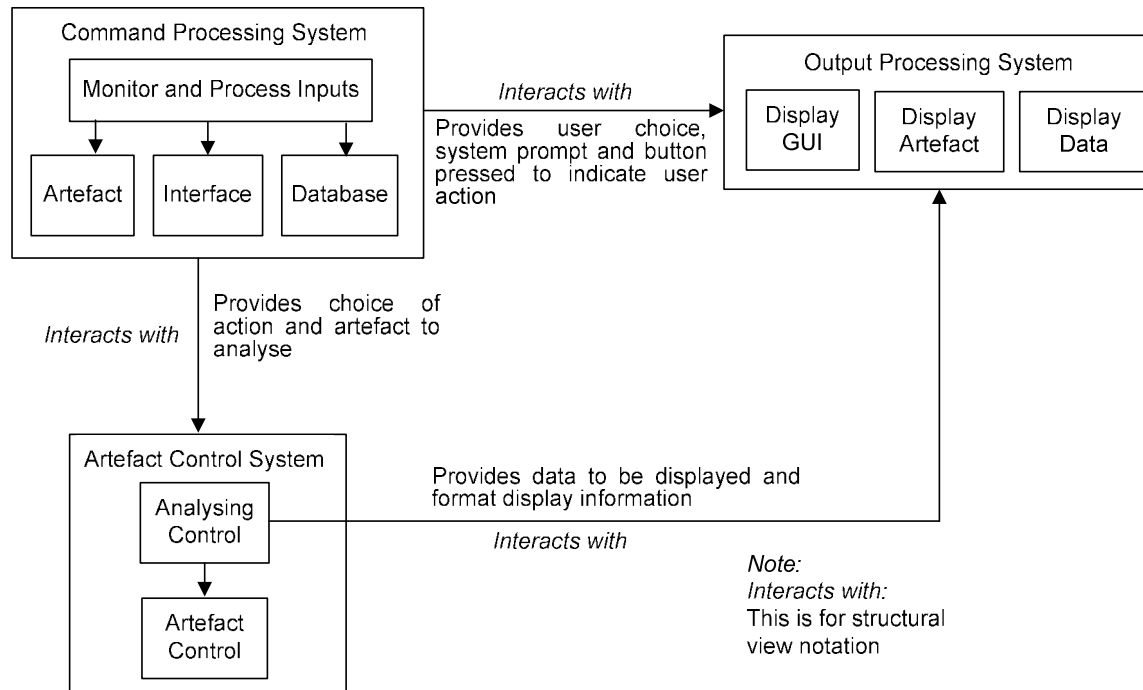


Figure 5.2: Structural and Data View of the Architecture

Table 5.1: Data Dictionary for the Structural and Data View Model Diagram

<u>Term</u>	<u>Description</u>	<u>Type</u>
Artefact	Processes input data containing artefact type for analysis task.	Sub-system
Interface	Processes input data containing user choice of action for analysis task.	Sub-system
Database	Processes input data containing sql statements.	Sub-system
Command Processing	Processes all inputs from the user.	System
Analysing Control	Processes output data from Command Processing System and controls system state based on those input.	Sub-system
Display GUI	Processes output data containing the format display for display on the GUI.	Sub-system
Display Artefact	Processes output data containing the type of artefact that is encoded for display on the GUI.	Sub-system
Monitor / Process Inputs	Processes pre-processed inputs and controls system state based on those inputs.	Sub-system
Output Processing	Generates output data that will be send to the GUI.	System
Artefact Control (sub-system)	Processes data sent from the Analysing Control sub-system and converts the data into the correct format for the model component.	Sub-system
Artefact Control (system)	Processes output data that can be used by the model components.	System

### 5.3.1 Classes Identified

As mentioned in Section 4.3 in Chapter 4, the package view of the architecture has four packages: entity domain package and entity business (entity data management) package; control package; boundary package; and database package. The entity classes that collaborate in the package are Artefact and Artefacts. The entity classes are in the package SAFTool::Entity::Domain. There is a need to be able to deal with the process of materialising instances of these classes from the database and, when required, materialising their links with other object instances or collections of object instances. The Broker pattern (see Section 5.3.4) is a way of making it possible to materialise the objects that are linked to other objects only when they are required. The brokers are in the package SAFTool::Entity::DataManagement, together

with any other necessary classes to handle the connection to the database.

The control classes in the SAFTool::Control package create the boundary classes. The control class needs to have the first dropdown menus populated with the names of all the artefacts, so it creates an instance of the control class ListArtefacts and requests it to pass back the artefact names to the boundary class, making reference to the boundary class in the message vadUI. The ListArtefacts instance sends the message addArtefactName (name) repeatedly to the boundary class until it has finished. It then returns control to the VisualiseArtefactData instance and destroys itself. The main control class can now enable the boundary classes, allowing the user to select a particular artefact. Instances of VisualiseArtefactDataUI need to be able to respond to the message addArtefactName (name) and realise the interface ArtefactLister.

Many other boundary classes will be needed to allow the user to select an artefact from the dropdown menu on the interface. Ideally the system should be able to reuse ListArtefacts in all the use cases where a list of artefacts has to be displayed in a boundary class. Interface is used to specify the operations that this entire boundary classes must respond to, that is the interface ArtefactLister. The boundary classes that need to display a list of artefacts.

In the SAFTool::Boundary package, the boundary classes AnalyseNewArtefactUI and VisualiseArtefactDataUI which handle the user interface will implement the ArtefactLister interface. An instance of the control class AnalyseNewArtefact is created first and that this creates a new instance of the AnalyseNewArtefactUI class to handle the user interface. This goes for the use case Visualise Artefact Data where VisualiseArtefactData is created first and this creates a new instance of the VisualiseArtefactDataUI class to handle the user interface. The sequence diagram presented in Figure 5.3 explains the packages and classes involved.

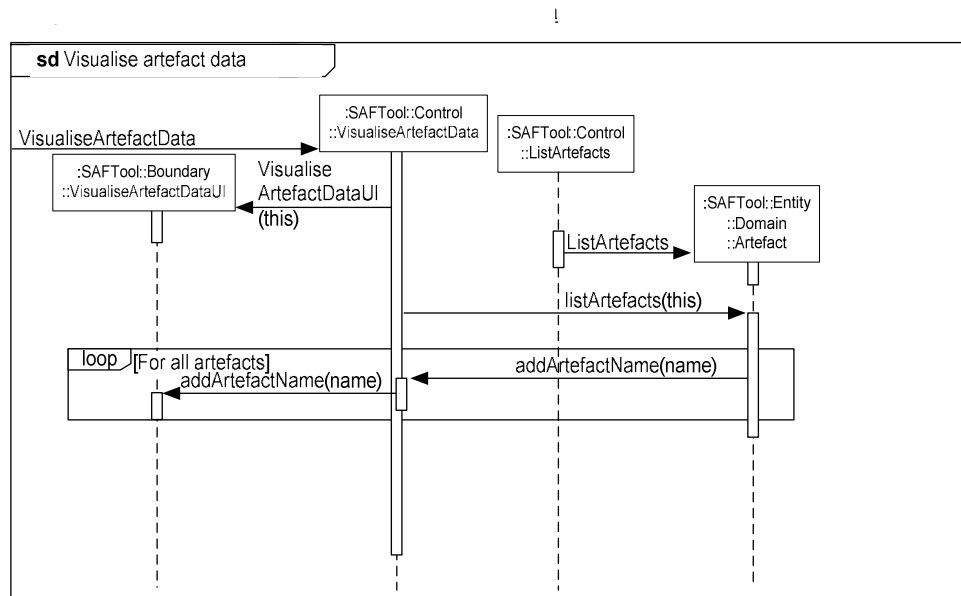


Figure 5.3: The Packages and Classes for the Windows System Generated Artefacts Forensic Analysis System

The class diagram for the architecture is shown in Figure 5.4. A full class diagram for the system can be found in Appendix E.

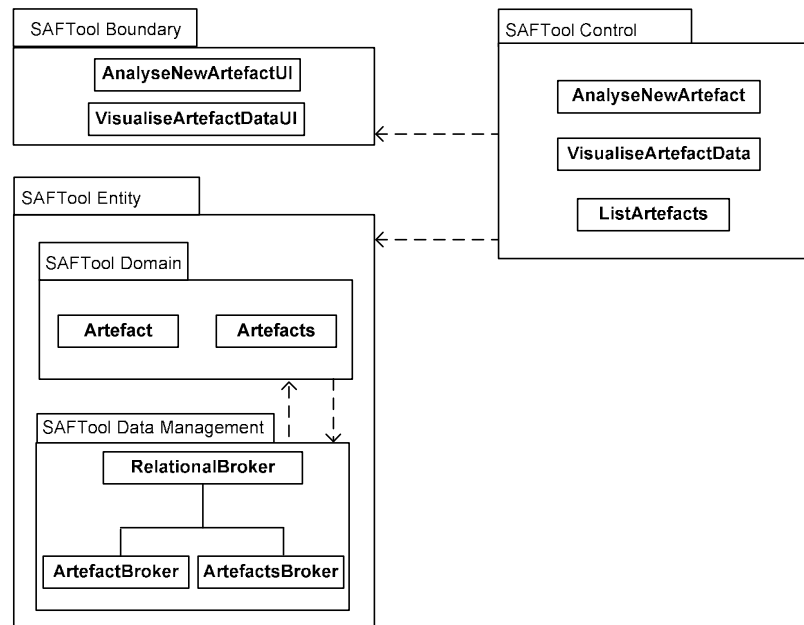


Figure 5.4: The Class Diagram for the Windows System Generated Artefacts Forensic Analysis System

### 5.3.2 Interaction and State Behaviour

An application in an object oriented technology application is a collection of objects. The behaviour of an object is what the object can do, each responsible for a small part of the system's overall behaviour. These behaviours are contained in the methods of the object, and these methods are invoked by a message that is send to it, and these objects produce the required behaviour through interaction, by exchanging messages that request information, that gives information or that asks another object to perform some task.

The sequence diagram that adds the boundary and control classes to a collaboration of the entity classes helps to show interaction occurrences. A detailed sequence diagram showing the various possible interactions can be found in Appendix D.

Users' primary objectives need to be taken into consideration when designing the user interface. In this project, the primary tasks are to recover data of possible forensic value from within an artefact and to visualise the artefact in order to display this information to a user. Figure 5.5 shows the interface of a Java program to implement the use case *Analyse New Artefact* for the Windows System Generated Artefacts Forensic Analysis System. In this use case, the user first selects the name of an artefact from a list box labelled *Artefact Name and Path*. At this point, no artefact is selected, and the user can click on the arrow at the end of the list box to view the list and select an artefact. When an artefact has been selected, the user can click on the button labelled *Start*. This interface works in a way that it can scan for the artefact file if the investigator does not provide the path to where the artefact file resides.

Requirements to locate the artefact and access the artefact for the architecture are realised by using the exporting features of the forensic tool that imaged the target media during a typical investigation. Using this forensic image to extract the artefacts makes them available for further analysis. The user must save or export the artefacts to a location that can be

accessed by the tool. The user can provide the path to where the exported artefacts now reside. If the user does not provide the path to where the artefacts reside, the tool can scan for the artefacts. The architecture checks for the present of the artefacts.

The tool will recover the data in the artefact and display the data contained in the artefact in a new window. In this interface design, the Start button has been disabled until an artefact has been selected. This method is used to ensure the reliability of the tool that is the resistance to failure of the system. State machine diagrams have been used to model the state of elements of the user interface in order to ensure that the behaviour of the interface has been specified correctly.

The sequence diagram and the prototype dialogue window developed in Appendix D do not indicate the permitted states of the interface. The sequence diagrams show only the sequential view of the user working through the fields on the screen from the top to the bottom, but it is in the nature of GUI interfaces that the user can click on interface objects out of sequence. A state machine or event action table has been used to model all these issues. Table 5.2 shows the event action table for the state machine of Figure 5.6 and outline the full range of actions that the user might initiate on the prototype.

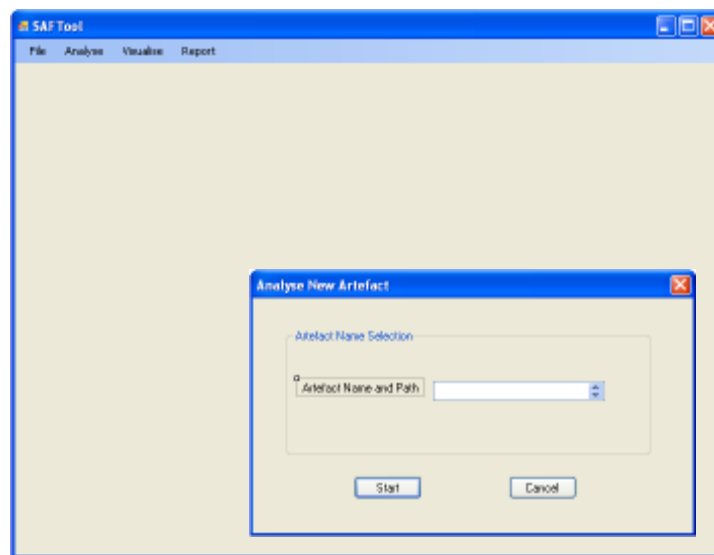


Figure 5.5: Interface for the Use Case Analyse New Artefact

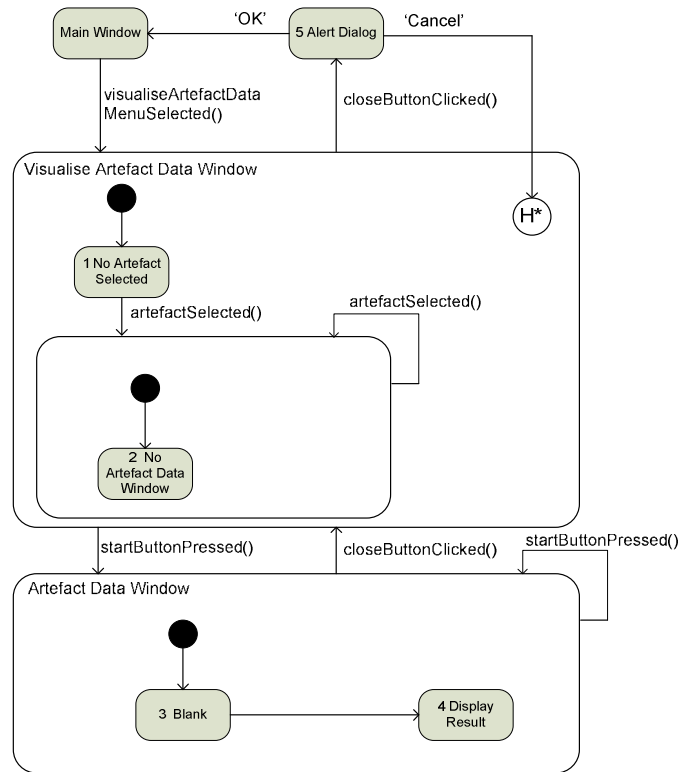


Figure 5.6: State Machine for the Visualise Artefact Data Window

Table 5.2: Event Action Table for Figure 5.6

Current State	Event	Action	Next State
-	Visualise Artefact Data	Display VisualiseArtefactDataUI Load artefact dropdown Disable start button Enable window	1
1	Artefact selected	Clear Artefact Data window Enable start button	2
2, 3, 4	Artefact selected	Clear Artefact Data window Enable start button	2
2	Artefact Data window displayed	Clear window panel Enable start button	3
3	Start button clicked	Display result	4
3, 4	Artefact Data window displayed	Clear window panel Enable start button	3
4	Start button clicked	Display result	4
1, 2, 3, 4	Close button clicked	Display alert dialogue	5
5	OK button clicked	Close alert dialogue Close window	-
5	Cancel button clicked	Close alert dialogue.	H*

### 5.3.3 Data Management and the Database

Data preparation involves making sure that all files and databases are ready to be used when the user wants to start using the system. During a typical investigation, a forensic image is created of the suspects' media. For this project, it is assumed that the forensic image was made of the suspects' media, and it is from this image thereof the Event log files and swap files were extracted by the investigator.

Typical forensic software applications offer a facility to copy or export files, allowing the investigator to extract particular files of interest for further analysis. Both the main forensic tools, FTK (AccessData, 2009), and EnCase (Guidance Software, 2005), offer this facility. To use the prototype tool developed as part of this thesis, the Event log files and Swap files the investigator wishes to examine must be exported to a location that can be accessed by the tool.

An alternative method to extracting the files from a forensic copy would be to use the Windows API and virtual machine to provide access to a virtual copy of the suspects system. This could use a live GUI manager in order for the API calls to be executed and retrieve the data. An example or a similar tool is Regedit.exe used to view the registry and Event log viewer used to view Event logs, both of these tools are supplied with the Windows operating system. After some consideration this method was discarded as it inhibited the flexibility of the architecture as it relied on the use of an API for access to the desired artefacts for analysis. The decision was therefore taken to extract the files, parse the data, and store the data for further examination.

With the information about the data structure of an artefact makes it a relatively straightforward process to parse the contents of an artefact, resulting in meaningful structure for the entries. Each artefact will have a different data structure, therefore, it is a requirement to have means to parse the various data structure and data types. For the not known data structure artefacts (swap file), the method will be every four megabytes of data from the



stream of bytes of data will be stored. This 4 is used due to, when the memory in used exceeds the amount of RAM available, the operating system will move pages (4KB pieces) of virtual address spaces to the swap file. 4MB is used instead of 4KB due to the size of the swap file used which averaged 555673KB. Plus, the stored data can be accessed in a read only manner, and this prevents making any changes to the original data.

So, these capabilities: organise and manage the tasks associated with storing and providing effective access to the data; enable a degree of flexibility in accessing the data; control the access to the data and permissions granted to different users for different levels of access; and the integrity of the data is maintained, make a DBMS the choice of this Windows System Generated Artefacts Forensic Analysis System and relational DBMS is appropriate because there are large volumes of data with varying access requirements for future reference (as mentioned in Section 4.8 and Section 5.2.3). The DBMS used by the application will be accessed from the programs using class libraries to provide the database access functionality and these class libraries are widely available for commercial DBMSs.

A relational DBMS is the most widely used type of DBMS (Digitivity, 2008). The relational database concept has been developed over a number of years and is robust, efficient and flexible for the kind of data it is designed to handle (Bennett et al., 2006). Examples of RDBMSs are Access, Oracle, SQL-Server, DB2, Informix, Ingres, Progress and Sybase. In Java, Apache Derby or Java DB is a relational database management system and open source database and is written in Java. The JavaDB database can be used to suit the storage of data for known and not known data structures of Windows system generated artefacts and can be used further for data from different systems.

By using the relational database, it is possible to extend the usage of the RDMBS to support various types of data for the internal structures of the artefact, for example, the Windows event logs and the swap files. To implement the JavaDB RDBMS for the Windows System Generated Artefacts Forensic Analysis system, this is built using NetBeans IDE, the following

must be developed in order for the system to store data from the internal structure of the artefacts:

1. A database design must be designed to deal with the internal data structure of the artefacts. This should show all the tables and fields of the internal structures in the database. For each internal structure, the design is the structure of the tables to use to represent classes in the database. It is only the attribute values of the object instances that are stored in an RDBMS, the operations are implemented in the programming languages itself.
2. Data is added to and retrieved from relational databases using SQL statements.
3. Support the further extension and development of the database to include additional tables for other artefacts. In NetBeans this can be accomplished by two ways: using external SQL script, and a tool when a table from another database to be recreated in the new database you are working with. The process used by the tool has two parts: copy the structure of the table definition of the table of the other database and then recreate the table in the new database.

In order to store the objects from an object oriented system in a relational database, the objects must be flattened out (Hautefeuille, 2011). The approach by which classes are converted to tables in a relational database is ‘flattening’ the classes into tables in order to design the storage structures. By mapping classes into tables, the collection classes for a set of objects of the same class are stored in tables. Selecting every row from the table is the way if it is necessary to iterate through every instance of a particular class. Objects do not have keys, so object identifiers are allocated to them and use an attribute that has a unique value of an instance of a class as a foreign key.

Figure 5.6 shows the database design for the Event log and Swap file structures. Each structure can be added to the database, using the Artefact for the database name. For example, in Figure 5.7 the Event logs table and the

Swap files table has a foreign key that is `ArtefactUID`. This is the attribute value of that table that denotes its structure type is Event logs or Swap files. The Event logs or the Swap files have only one table associated with their structure. For the Event logs, the event records in the files are saved to the database for further analysis. Each event log category has its own table in the database with its file name and timestamp for the table name. The purpose of choosing a file name and timestamp for the table name is to make it unique and retrievable for further analysis. As the user does need to know the table name and where the table is stored, the prototype tool automatically shows the artefact name and the user simply selects the desired artefact to be analysed.

This method has been demonstrated by Mee (2009) as an effective approach for storing information relating to system artefacts. Mee (2009) addressed the database design required to store elements of the Registry by introducing comparable underlying tables in which to store the data. The underlying tables were the Hive table, Key table and KeyData table. These dealt with the nested keys, and subkeys maintaining the hierarchy of the Registry data (keys may contain one or more subkeys).

The database design for this should be able to cater for each of the system artefact's different internal structures. The database can be designed and developed to suit different possible representations of the data, either linearly or hierarchically. In the case of mapping an inheritance hierarchy, the method used is to implement all the classes, both superclass and subclasses as separate tables. To retrieve the data for a subclass, both its own table and the table of its superclass is accessed. Each internal structure of a system artefact must go through this design as it is added to the database. The end user does not need to know the structure of the internal structure but the developers of the architecture must be able to identify and recognise the underlying structure in order to break the structure down to a database design. A full diagram of the database structure, including various internal structures, can be found in Appendix F which illustrates the database as a whole, including other system artefacts.

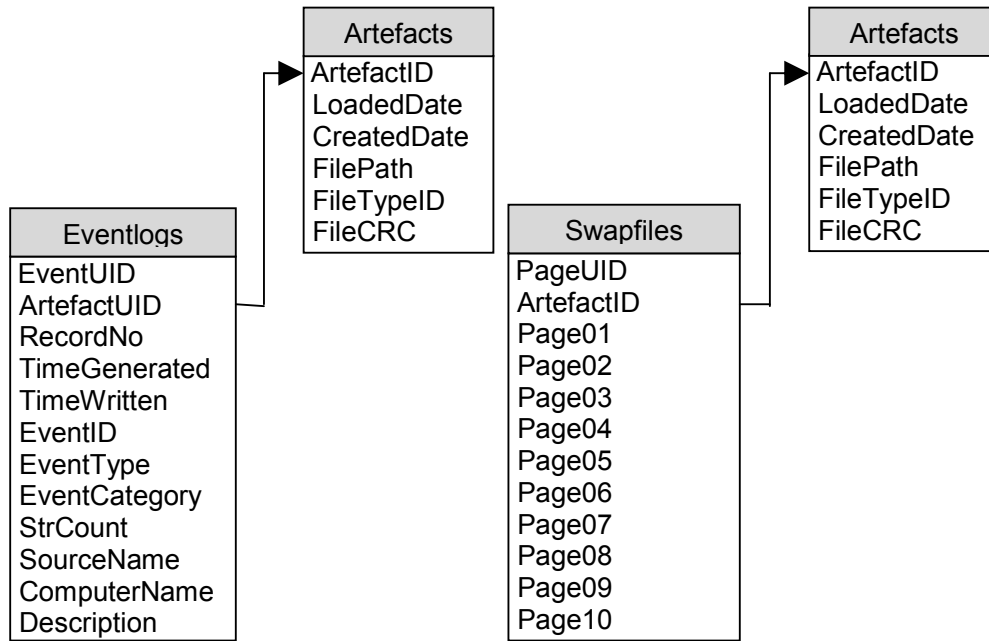


Figure 5.7: Database Design for the Event Logs and Swap Files Structures

### 5.3.4 SQL Statements

Each table in a relational database is made up of rows of data. Each row contains attribute values that are organised into columns. Each column contains data values of the same attribute type. Once the database design has been determined, the tables must be created in the database. Each attribute value in the table must be atomic, that is, it may not contain multiple values or be capable of being broken down further. Using the SQL Create statement, the tables can be inserted into the Database. Figure 5.8 below denotes the Create statements for the Artefacts and Event logs.

```
CREATE TABLE Artefacts
(
  ArtefactID int NOT NULL PRIMARY KEY,
  LoadedDate datetime NOT NULL,
  CreatedDate datetime NOT NULL,
  FilePath varchar(200),
  FileType int NOT NULL
  FileCRC varchar(200)
);

CREATE TABLE Eventlogs
(
  EventUID int NOT NULL PRIMARY KEY,
  ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
  RecordNo int NOT NULL,
  GeneratedTime datetime NOT NULL,
  WrittenTime datetime NOT NULL,
  EventID int NOT NULL,
  EventType int NOT NULL,
  EventCategory int NOT NULL,
  StrCount int NOT NULL,
  SourceName varchar(200) NULL,
  ComputerName varchar(200) NULL,
  Description varchar(200) NULL
);
```

Figure 5.8: SQL to Create Tables for the Artefacts and Event Logs

Using JDBC to interact with a database makes creating a database independent application possible. This means another database can be used without changing anything in the application. To implement the JDBC for this system, RelationalBroker abstract class was used. Figure 5.9 shows associations between RelationalBroker abstract class and classes from other package for the Windows System Generated Artefacts Forensic Analysis System.

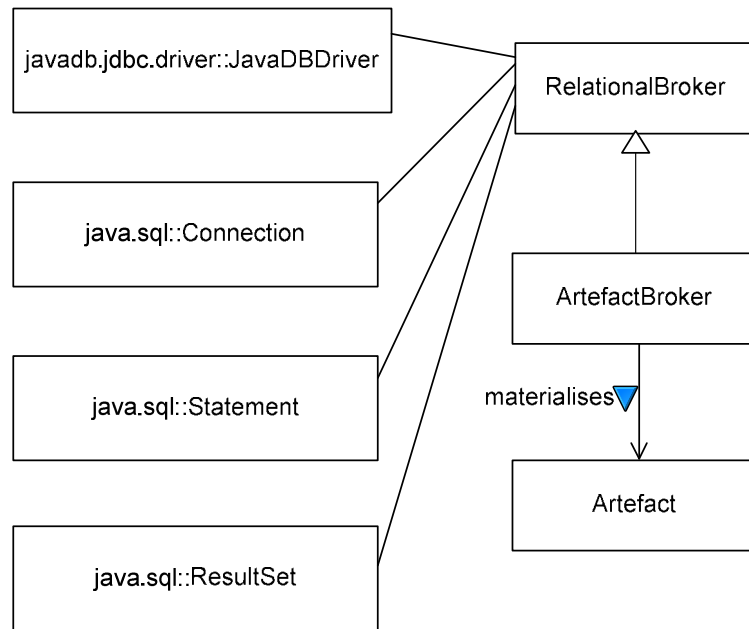


Figure 5.9: Associations Between `RelationalBroker` Class and Classes from Other Packages

### 5.3.5 Fulfilling the Architecture's Other Requirements

Chapter 4 contained a discussion of digital forensics unique requirements and limitation of the current state of the art tools that are used as the requirements of the architecture (Chapter 3). Such review and discussion revealed that there are other requirements that the architecture must comply with.

According to Walden (2007), *"the process of obtaining forensic data is a significant technical challenge for investigators, since it may modify the source data or its related metadata, fatally undermining the evidential value of the forensic material"*. A requirement for the architecture was therefore that the data stored would be deemed to be forensically sound data. This security requirement for the architecture is mentioned in Chapter 4. In order to implement this requirement, the MD5 message digest was calculated and stored in the database. MD5 is a computer algorithm that produces unique mathematical representations of the data through the creation of a 128 bit

message digest from the data input. The MD5 is calculated for both the original and the copy and then compared to verify data integrity (Newman, 2007). At any stage to ensure the file has not been changed, the MD5 hashing algorithm can be applied again and the resulting hash compared to the original to prove the file has not been altered. If the file has been changed, the MD5 hash values will not match, thus alerting users that the files have been changed. This technique is used daily in computer forensic practice to ensure file integrity.

According to O’Conner (2007), “*An extensible application is one that can be extended easily without modifying its original code base. Its functionality can be enhanced with new plugins or modules*”. A plugin is an assembly that registers itself with the main application framework and provides additional functionality through a defined interface. The main application framework will become the API that loads and runs plugins. The plugins communicate to the services through the main application framework API.

It was proposed that the architecture be extensible, that it is easy (supported by the design and additional documentation) to add additional support for new types of data sources regarding analysis. This is critical since new operating systems and applications are being continually developed and the system ideally has to be able to support these new sources with as effort as possible.

In this extensible architecture, new functionalities or application programming interfaces (APIs) can be added by simply adding a new Java Archive (JAR) file onto the application classpath or into an application specific directory. The JDBC is an API to unify work with any database. A developer just has to change the JDBC driver, which is the implementation of the JDBC API, and load the proper class.

The architecture allows others to upgrade or enhance specific parts of a tool without changing the core application. By using applications with extensible services, the architecture allows the user to provide service implementations. Service implementations are developed as plugins that can

be installed and started during runtime which require no modifications to the original application. The architecture allows the user to add new providers to the classpath or runtime extension directory for the ServiceLoader class to find it. ServiceLoader is an API that helps to find, load and use service providers. The reason why ServiceLoader class is used over the use of other module systems was due to the building of modular application and much more maintainable application is achievable. The service providers are registered by a configuration file in the JAR file's directory. Figure 5.10 shows the plugin framework. There are a number of services that provide different types of functionality to the plugins.

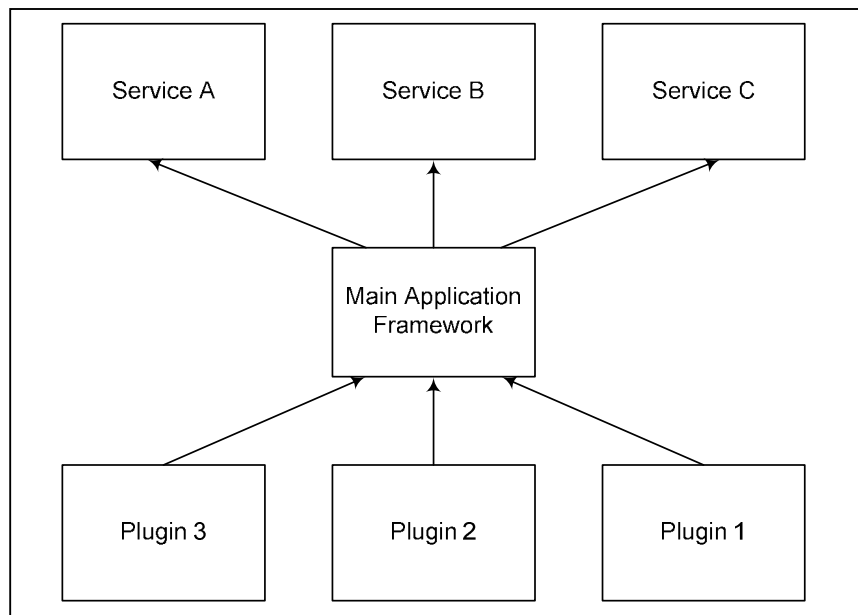


Figure 5.10: Plugin Framework

According to Fry (2007), *“whenever we analyse data, our goal is to highlight its features in order of their importance, reveal patterns, and simultaneously show features that exist across multiple dimensions”*. According to Teerlink and Erbacher (2006), *“using visualisation techniques to display information about computer data can help forensic scientists direct their searches to suspicious files”*. According to Guillermo et al. (2007) *“the use of visual representations to accelerate insight into complex data is a characteristic*



*of visual analytic software. Visual representations translate data into a visible form that highlights important features, including commonalities and anomalies”.*

Chapter 4 (Section 4.3) states the visualiser subsystem will deal with visualising the data that is retrieved in an intuitive format to enable investigators to extract evidence. Various techniques of visualisation include using illustrations of data such as timeline, data map, tree map, flow charts, graph and table. For example, if an investigator wishes to visualise data by chronological order, a plugin can be used to list the specified data for time stamps, and present the resulting to the user. It was proposed that the architecture present the data extracted from the artefacts’ internal structure visually. The architecture uses a plugin to graphically represent the data contained in the system. The plugin can represent the data in various formats tailored to the nature of the artefact. Other plugins can provide visualisation of data in a different manner to provide alternative ways to explore the data.

According to Nelson et al. (2004), *“a report is to communicate the results of computing forensics examination or investigation. A formal report presents evidence as testimony in court, at an administrative hearing, or as an affidavit. Besides presenting facts, reports can communicate expert opinion”*. It is proposed that the architecture provide the investigator with a reporting feature. Since the architecture supports various plugin, a reporting plugin can therefore be used to export out the data to a report. This will allow the investigator to render their findings and information into a report (various tagged pieces of data, visual representation of data, correlated search results) and also allow the investigator to add in various notes and other data related to the case, thus compiling a complete report that could be presented alongside expert witness affidavits. Table 3.23 in Section 3.4 illustrates the limitations of existing tools. It also highlights that both commercial tools discussed in Chapter 3 supports a reporting facility for the user.

## 5.4 Implementation Diagrams

According to Finkelstein (2000), “*Software engineering focuses on the real world goals for, services provided by, and constraints on such system; the precise specification of system structure and behaviour, and the implementation of these specifications; the activities required in order to develop an assurance that the specifications and real world goals have been met; the evolution of such systems over time and across system families*”.

The implementation diagrams illustrate the physical implementation, that is, the components structure and the deployment for the runtime. There are two diagram types associated with implementation diagrams: the component diagram and the deployment diagram. The thesis had the objective of developing a multiplatform architecture, as such multiplatforms promote the portability of the architecture.

The component diagram's shows the structure of components with its relationships of a system. In the component diagram, the components of an application or system, interactions with their interrelationships, and their public interfaces are depicted. According to Ambler (2003), “*UML component diagrams are great for identifying the architectural landscape for your system as they enable you to model the high level software components, and more importantly the interfaces to those components*”. Figure 5.11 shows the component diagram for the Windows System Generated Artefacts Forensic Analysis System with the components wired together to form larger components and the connection between the internal subcomponents. The diagram shows the different components, such as Artefacts and Artefact in the Model layer and how the Controller layer component interacts with these components. The diagram also depicts a database access component that represents a library component that the Model layer components will use to interact with a database.

The deployment diagram shows the run-time architecture of a system.

Each node represents either a physical machine node or a virtual node and the connections between the nodes show the system interactions paths. An example of a node is a mainframe node. The deployment diagram involves modelling the hardware configurations together with the software components.

Nowadays, software applications are complex in nature. Software applications can be stand alone, web based, distributed, mainframe based and so on. The application is assumed to be a desktop system based application which is in a standalone environment: deploys services locally, uses the services, and terminates the services when they are no longer needed. Services locally deployed by this application are not available to any other application, that is, no remote services are available.

The nodes for the deployment environment and the relationships among them are as follows:

- Application Server represents the computer that will receive and process user requests and send responses from the application. This node consists of different components of the Windows System Generated Artefacts Forensic Analysis System, such as View, Controller, Model, and Database Access.
- Database Server represents the node that hosts the database server. This node used to store and retrieve the data by the Windows System Generated Artefacts Forensic Analysis System components.

Figure 5.12 shows the deployment diagram for the Windows System Generated Artefacts Forensic Analysis System. The deployment diagram shows the two nodes SAFTool\_APP\_SERVER and SAFTool\_DB\_SERVER that represent the nodes application server and database server, respectively. The View, Controller and Model components are depicted in SAFTool\_APP\_SERVER node and the three are interconnected in the Windows System Generated Artefacts Forensic Analysis System.

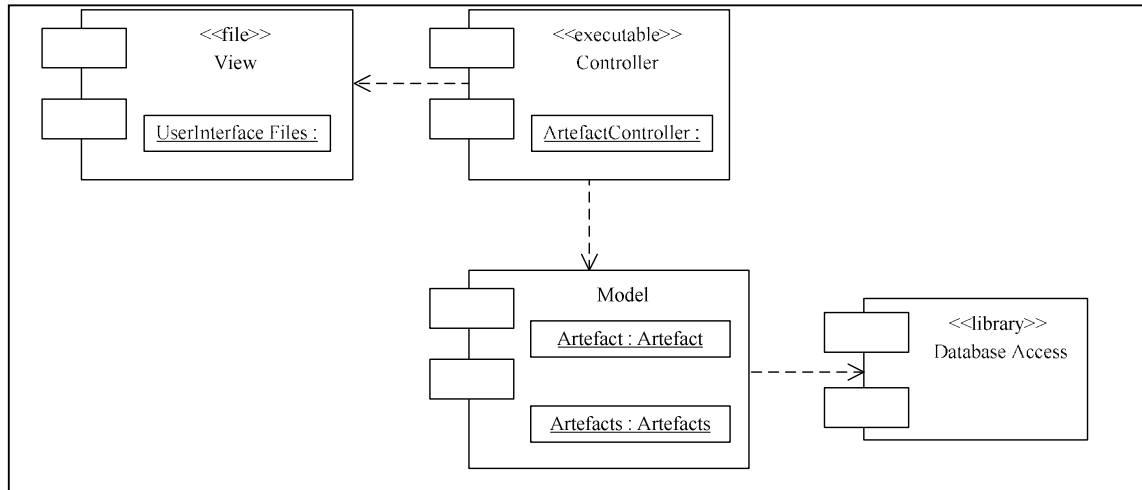


Figure 5.11: Component Diagram for the Windows System Generated Artefacts Forensic Analysis System

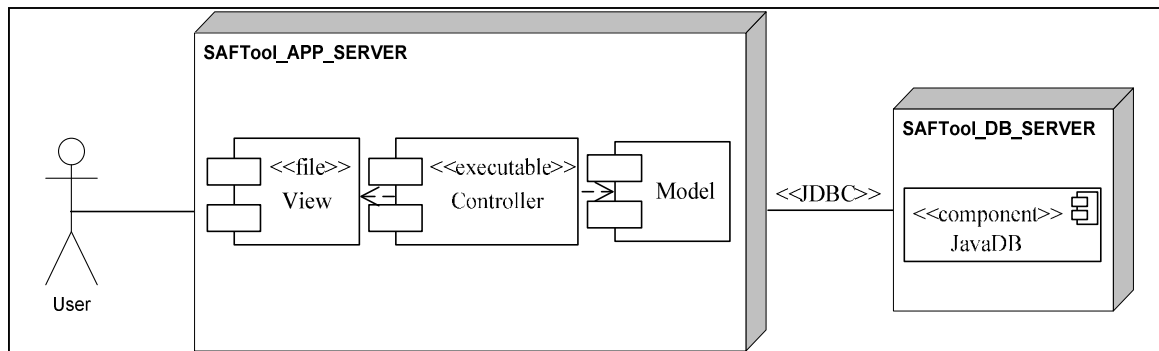


Figure 5.12: Deployment Diagram for the Windows System Generated Artefacts Forensic Analysis System

## 5.5 Conclusion

In this chapter, the implementation of the architecture was discussed and the justification for the tools and languages used was outlined. Further, the products used and created by the design workflow and implemented in the architecture seek to address the limitations of the current state-of-the-art tools.

The hypothesis of the thesis was to develop an architecture that is able to integrate forensic data from known and not known internal data structures of system generated artefacts by the Windows operating system and to design and implement a proof of concept prototype tool, with appropriate example artefacts with key features of the proposed architecture are: extensible, flexible and multiplatform to improve forensic process. Considering each of the proposed packages of the architecture, they only appear to meet the main criteria of the architecture: extensibility; flexibility; and multiplatform implementation.

The implementation of the tool will enable the assessments outlined in Chapter 6 to be conducted and the hypothesis proposed in Chapter 1 to be tested (Section 1.3).

## CHAPTER 6

# EVALUATION AND DISCUSSION

This chapter presents the results of the evaluation methods applied to the system prototype and the proposed architecture.

As illustrated in the Table 6.1, it is evident that the architecture works, and that it satisfies all functions and the hypothesis of the thesis. The next section will deal with the evaluation of the architecture with a comparison of the use of other tools.

The first evaluation analyses the results obtained from using the prototype tool to analyse nine different Event log files and nine Swap files. The analysis involves the correct parsing of system artefacts through the process of extracting data structures and visualising the results in the forms of narrative constructs and also in graphical form. The output from the system is compared to that of other available open source forensic software.

The second evaluation provides further empirical data relating to how well the system supports the analysis of Windows system generated artefacts through experts evaluation. The purpose of the empirical studies of the forensic analysis of Windows system generated artefacts was not to evaluate

the quality of system generated artefacts used as digital evidence, but to evaluate the functionalities and usefulness of the prototype software and to establish if it successfully assists the investigator to determine what data is available within the Windows operating system generated artefacts.

Table 6.1: Requirements of the Architecture Revisited  
Illustrating the Architecture Satisfies Each

Function	Architecture and System	
Facilitate the analysis and visualisation of forensic data in various types of file format and data complexity.	✓	Section 5.3.2
Locate the artefact files for analysis.		Section 5.3.2
Facilitate data extraction, data interpretation, and data reconstruction from the internal structure of the artefact.	✓	Section 5.3.3
Visual representation of data from the data structures whether by graphs, charts or illustrations.	✓	Section 5.3.5
Reporting facility	✓	Section 5.3.5
Scalability: Facilitate the extending of supporting new kinds of kinds of target and new types of analysis.	✓	Section 5.3.5
Data integrity: No data can be changed while being stored in the system.	✓	Section 5.3.3
Data integrity: No data in the original complex structure of the evidential artefact will change.	✓	Section 5.3.3
Verifiability: Uses Hash Value for individual file.	✓	Section 5.3.5
Other Requirement: The architecture and implementation should aim at being as independent of underlying software as possible.	✓	Section 5.2.2
Usability: Easy navigation, easy to use by using keyboard and mouse to control the system.	✓	Section 5.3.2
Other Requirement: Authentication software is used to prove that the evidence has not been changed. Algorithms like MD5 or SHA-1 are required.	✓	Section 5.3.5
Other Requirement: The tool must be designed in such a way that it provides a form of integrity assurance or record when its utilities are executed, such as it provides an audit record about timestamp or actions taken, or results return from running those utilities. This requirement needs to be addressed, as this is compliant with the best practice requirements for forensics tools as discussed in Chapter 2.	✓	Section 5.3.3
✓ - denotes the presence of the function and requirement		

## 6.1 Research Evaluation

In Chapter 1 an experimental method for the research project was proposed. In this section the research methodology for the evaluation of the architecture and the prototype system is examined. According to Kumar (as edited in Rutman, 1977) *“Evaluation research is, first and foremost, a process of applying scientific procedures to accumulate reliable and valid evidence in the manner and the extent to which specific activities produce particular effects or outcomes”*.

In order to assess the proposed method to visualise information extracted from Windows system generated artefacts, a number of different approaches were selected. Experiments are proposed to measure the accuracy and questionnaires are prepared to verify functionality of the prototype. An experiment is a process or study that results in the collection of data. The results of experiments are not known in advance (SAS, 2010). According to Kumar (2005), *“A questionnaire is a written list of questions, the answers to which are provided by respondents”*. The questionnaire comprised questions designed to assess the prototype’s ability to process the selected system artefacts. An assessment of the prototype’s functionality against that of other open source tools was also carried out. Finally, a selection of expert opinion was sought on the key aspects of the architecture and prototype implementation. The detailed experimental setup is discussed in Section 6.5.

A clear definition of the details of an experiment makes the desired statistical analyses possible, and almost always improves the usefulness of the results. Furthermore, the objective of designed experiments is to improve the precision of the results in order to examine the research hypotheses (SAS, 2010). The data collection and analysis plan are very important to meet the specific objectives of an experiment. The data collection and analysis plan statements for this research were as follows:

- i. The experiment with swap files and event log files sourced from the ongoing disk study work at the University of Glamorgan.



Performance of the prototype software was evaluated through a series of experiments on a selection of Event logs and Swap files obtained from the forensic lab in which forensic images were taken during a disk study. A selection of Event logs and Swap files were selected among the readable hard disks images from different countries. These two types of system generated artefacts have different file sizes and come from different versions of the operating system. Furthermore, the Event logs and Swap files came from a large number of disks that have been purchased in a number of countries without knowing the background of their sources (Jones et al., 2009).

- ii. A comparison with current state-of-the-art tools.

A comparison of the proposed system was to be made against current state-of-the-art tools used. Performing the same task using different tools can be used to verify the results (Nelson et al., 2004).

The objective of this assessment was to obtain the right results with the relevant system artefacts and quantify the accuracy of parsing correctly organised records of system artefacts through the process of extracting known and not known data structures and visualising them in the forms of narrative construct and statistical graph.

- iii. A review of the key design features of the prototype.

Besides quantifying the accuracy of the data extracted, verifying the functionalities and usefulness of the prototype software was done through eliciting expert opinion. The objective was to verify the ability of the implemented prototype software to visualise the known data structure of Event logs and the not known data structure of Swap files in such a way that the investigator could easily see what data was available within these system areas of the Windows operating system. It was also to verify that the architecture is extensible for extracting information from the Event logs and Swap files and to allowing some plug in features to be added to it such as visualisations and various forensic analysis capabilities. A qualitative review of the input from experts in the field of forensics would aid the assessment of the system's functionality.

The experimental setup for accomplishing the data collection and analysis plan is detailed in Section 6.1.2 and the results are recorded as evaluation measurements.

### **6.1.1 Data Interpretation and Analysis**

The results gained from the experiments were used to compare the developed software application with that of other tools. To date, there are no published evidence concerning experiments in analysing and presenting data for Windows system generated artefacts. Hence, the accuracy of the developed software application was compared with that of tools used for data extraction, i.e. Event Viewer and Carvey's (2007) tools (evtstats.pl and lsevt.pl) for the Event logs and the WinHex for the Swap files.

Experiments were performed to investigate the number of records parsed and the total size of data parsed; and processing time taken in parsing the contents of the files. These experiments were used to measure the accuracy scores of the data extracted and the time taken to process this data. Measuring the time taken to process the data is due to that informational visualisation would normally be reached after much data mining in the analytical tier. As well as quantifying the accuracy of the data structure extracted, verifying the functionalities and usefulness of the prototype software was achieved through questionnaires eliciting experts' opinions. The results from the comparative study and the questionnaires were analysed using the Microsoft Excel statistical package for data analysis and results presentation.

### 6.1.2 Experimental Setup

#### Test Data

To evaluate the prototype software (and to be able to make statements concerning the suitability of the architecture) a selection of test data was required. The test data consists of known data structure (event logs file) and not known data structure (swap file) system generated artefacts. This test data extracted from the Windows XP and Windows Vista operating systems was obtained from hard disk images that had been created as part of a disk study into data disposal practises at the University of Glamorgan Computer Forensics Research Laboratory (Jones et al., 2009).

The test data preparation process was as follows:

1. Hard disk images were selected among the readable disks from different countries. Only 10 hard disk images contained viable event logs files and swap files to be used as test data. Some of the hard disks had been deleted or corrupted and were unusable for this study. For some of the readable disks, attempts were made to remove the data from the disks by deletion, formatting or reinstallation of the operating systems.
2. From 10 hard disk images, the following were extracted:
  - a. 12 binary format of event log files; and
  - b. 10 binary format of pagefile files.
3. The 12 binary formats of event logs and the 10 binary formats of pagefile files were numbered as in Table 6.2.

Table 6.2: Numbering for Event Logs Files and Swap Files

No.	Event log file Name
1.	AppEvent01.Evt
2.	SecEvent01.Evt
3.	SysEvent01.Evt
4.	AppEvent02.Evt
5.	SecEvent02.Evt
6.	SysEvent02.Evt
7.	AppEvent03.Evt
8.	SecEvent03.Evt
9.	SysEvent03.Evt
10.	AppEvent04.Evt
11.	SecEvent04.Evt
12.	SysEvent04.Evt

No.	Swap file Name
1.	pagefile01.sys
2.	pagefile02.sys
3.	pagefile03.sys
4.	pagefile04.sys
5.	pagefile05.sys
6.	pagefile06.sys
7.	pagefile07.sys
8.	pagefile08.sys
9.	pagefile09.sys
10.	pagefile10.sys

## Experiment 1

The goal of the experiment was to evaluate the results after the developed software application had been tested to determine whether it was functioning as desired under normal conditions.

The experimental process was carried out in the following steps:

1. Use AppEvent04.Evt with Prototype Software.

Measurement used:

- Successful operation
- Unsuccessful operation

2. Use SecEvent04.Evt with Prototype Software.

Measurement used:

- Successful operation
- Unsuccessful operation

3. Use SysEvent04.Evt with Prototype Software.

Measurement used:

- Successful operation
- Unsuccessful operation

4. Use pagefile10.sys with Prototype Software.

Measurement used:

- Successful operation
- Unsuccessful operation

## **Experiment 2**

The goal of the experiment was to evaluate the results of the prototype software application against those produced by other tools. The prototype tool needed to be examined in terms of its effectiveness in dealing with not known data structure (swap file) and known data structure (event log) files. The tools selected for this comparison were the Carvey (2007) Perl script (evtstats.pl) and Event Viewer for the event log files. evtstats.pl was used as this script parses through the Event log files in binary mode, bypassing the Windows API altogether, by doing it this way the Event log files can be parsed on a platform other than Windows, although Event Viewer gives error messages that a file is somehow corrupted. Event Viewer is a GUI manager for the Event logs and is available from Microsoft as a component of the operating system which interprets the event data and displays it in a readable fashion. The various ways in which the user can display; work with; and place conditions on an event is a real benefit and getting used to it is relatively easy.

For the pagefile files, the tool selected for this comparison was the WinHex application. The WinHex application was used as it provides a

convenient and simple interface for random access to files and disks at the sector level through its hexadecimal editor in analysing files.

A. For Event Log files

1. Using evtstats.pl

This experiment was conducted to retrieve and examine the data using evtstats.pl

- i. This experiment used 9 binary format of event log files (AppEvent01.Evt until AppEvent03.Evt, SecEvent01.Evt until SecEvent03.Evt and SysEvent01.Evt until SysEvent03.Evt) described in Table 6.2 to run with evtstats.pl to collect information from each event log files.
- ii. evtstats.pl displays simple statistics for each event log files as shown here:

- Max size of Event Log file =
- Actual size of the Event Log file =
- Total number of event records (header info) =
- Total number of event records (actual count) =
- Total number of event records (rec\_nums) =
- Total number of event records (sources) =
- Total number of event records (types) =
- Total number of event records (IDs) =

iii. Statistic used:

Total number of event records (actual count) =  $Xn$ , where  
 $n = 01, 02, 03$

2. Using Event Viewer

This experiment was conducted to retrieve and examine the data using Event Viewer.

- i. This experiment used 9 event log files (AppEvent01.Evt until AppEvent03.Evt, SecEvent01.Evt until SecEvent03.Evt and SysEvent01.Evt until SysEvent03.Evt) described in Table 6.2 to run with Event Viewer to ascertain how many records are present in each event log file.

- ii. Statistic used:

Total number of event records displayed =  $Yn$ , where  
 $n = 01, 02, 03$

3. Using Prototype Software

This experiment was carried out to retrieve and examine the data using the Prototype Software.

- i. This experiment used 9 event log files (AppEvent01.Evt until AppEvent03.Evt, SecEvent01.Evt until SecEvent03.Evt and SysEvent01.Evt until SysEvent03.Evt) described in Table 6.2 to run with the Prototype Software and obtain the number of records present in each event log file.

- ii. Statistic used:

Frequency of event records display =  $Zn$ , where  
 $n = 01, 02, 03$

4. Results

The experimental results are presented in tabulated form below:

Table 6.3: Event Logs Experimental Results for Experiment 2

Test Data File Name	No. of Records		
	evtstats.pl	Event Viewer	SAFTool
AppEvent01.Evt	$Xn$	$Yn$	$Zn$
SecEvent01.Evt	$Xn$	$Yn$	$Zn$
SysEvent01.Evt	$Xn$	$Yn$	$Zn$
AppEvent02.Evt	$Xn$	$Yn$	$Zn$
SecEvent02.Evt	$Xn$	$Yn$	$Zn$
SysEvent02.Evt	$Xn$	$Yn$	$Zn$
AppEvent03.Evt	$Xn$	$Yn$	$Zn$
SecEvent03.Evt	$Xn$	$Yn$	$Zn$
SysEvent03.Evt	$Xn$	$Yn$	$Zn$
$Xn, Yn, Zn = \text{Number of Records}$ where $n = 01, 02, 03$			

## B. For Swap Files

## 1. Using WinHex

This experiment was conducted to retrieve and examine the data using the WinHex.

i. This experiment used 9 pagefile files (pagefile01.sys until pagefile09.sys) as described in Table 6.2 with the WinHex to collect information from pagefile files.

ii. The statistic used:

Existence of 'mail', 'from', 'www', 'html' and 'send' words found  
 $= AN_n$ , where

$AN_n = Yes / No$ , where

$N = V, W, X, Y, Z$

$n = 01, 02, 03, 04, 05, 06, 07, 08, 09$

## 2. Using Prototype Software

This experiment was carried out to retrieve and examine the data using the Prototype Software.

i. This experiment used 9 pagefile files (pagefile01.sys until pagefile09.sys) as described in Table 6.2 with the Prototype Software to collect information from pagefile files.

ii. Existence of 'mail', 'from', 'www', 'html' and 'send' words found  
 $= BN_n$ , where

$BN_n = Yes / No$ , where

$N = V, W, X, Y, Z$

$n = 01, 02, 03, 04, 05, 06, 07, 08, 09$

## 3. Results

The experimental results are presented in tabulated form below:



Table 6.4: Swap File Experimental Results for Experiment 2 Using WinHex

File No.	Existence of 'mail', 'from', 'www', 'html' and 'send' words in WinHex (ANn)				
	'mail'	'from'	'www'	'html'	'send'
pagefile01.sys	AV01	AW01	AX01	AY01	AZ01
pagefile02.sys	AV02	AW02	AX02	AY02	AZ02
pagefile03.sys	AV03	AW03	AX03	AY03	AZ03
pagefile04.sys	AV04	AW04	AX04	AY04	AZ04
pagefile05.sys	AV05	AW05	AX05	AY05	AZ05
pagefile06.sys	AV06	AW06	AX06	AY06	AZ06
pagefile07.sys	AV07	AW07	AX07	AY07	AZ07
pagefile08.sys	AV08	AW08	AX08	AY08	AZ08
pagefile09.sys	AV09	AW09	AX09	AY09	AZ09

The results of the experiment will be entered in the form:

$ANn = \text{Yes} / \text{No}$ , where

$N = V, W, X, Y, Z$  and

$n = 01, 02, 03, 04, 05, 06, 07, 08, 09$

Table 6.5: Swap File Experimental Results for Experiment 2 Using Prototype Tool

File No.	Existence of 'mail', 'from', 'www', 'html' and 'send' words in Prototype Software (BNn)				
	'mail'	'from'	'www'	'html'	'send'
pagefile01.sys	BV01	BW01	BX01	BY01	BZ01
pagefile02.sys	BV02	BW02	BX02	BY02	BZ02
pagefile03.sys	BV03	BW03	BX03	BY03	BZ03
pagefile04.sys	BV04	BW04	BX04	BY04	BZ04
pagefile05.sys	BV05	BW05	BX05	BY05	BZ05
pagefile06.sys	BV06	BW06	BX06	BY06	BZ06
pagefile07.sys	BV07	BW07	BX07	BY07	BZ07
pagefile08.sys	BV08	BW08	BX08	BY08	BZ08
pagefile09.sys	BV09	BW09	BX09	BY09	BZ09

$BNn = \text{Yes} / \text{No}$ , where

$N = V, W, X, Y, Z$  and

$n = 01, 02, 03, 04, 05, 06, 07, 08, 09$

### **Experiment 3**

The goal of the experiment was to evaluate the results after the developed software application had been tested in a controlled environment to determine whether it was functioning effectively. The experimental process was carried out according to the following plan:

#### **Subjects of Experiment**

The subjects were recruited from among post-graduate students in the Information Security Research Group, University of Glamorgan.

#### **Structure of Experiments**

##### **i. Data**

The experiments used AppEvent04.Evt, SecEvent04.Evt, SysEvent04.Evt and pagefile10.sys from the examined disks (readable disks) that were used in the analysis of information remaining on disks obtained as part of the disk study at the University of Glamorgan.

##### **ii. Questionnaires:**

A questionnaire (see Appendix I) was given to all subjects to be answered after completion of the set task.

##### **iii. The process:**

The process of the questionnaire is carried out by giving the participants briefing describing the tasks and objectives of the questionnaire.

**iv. The tasks:**

The following tasks were used in the experiments:

**A. For the Extensible Architecture of the Prototype**

Task 1: To verify the extensible architecture.

**B. For Event log files**

Task 1: To install the event log plugin with SAFTool.

Task 2: To open and analyse the event log file with SAFTool.

Task 3: To open and further analyse the event log file (visualise event log files).

Task 4: To generate report with SAFTool.

**C. For Swap files**

Task 1: To install the pagefile plugin with SAFTool.

Task 2: To open and analyse the pagefile file with SAFTool.

Task 3: To open and further analyse the pagefile file (visualise pagefile file) with SAFTool.

Task 4: To search word (keyword search) with SAFTool.

Task 5: To generate report with SAFTool.

## 6.2 Comparison with Other Tools

The Windows system generated artefacts forensic analysis system concerns the use of data structure (refer to Definition 2 of Chapter 1) when extracting data of an artefact and visualising the data. The proposed tool for this purpose is similar to other tools related to forensic analysis such as JView (Microsoft, 2005), EnCase Registry Viewer (Guidance Software, 2005), FTK Registry Viewer (AccessData, 2009). However, these tools in particular EnCase (Guidance Software, 2005); and FTK (AccessData, 2009) focus on the Windows Registry as the key Windows artefact. In addition, comparing this approach to (Vlastos and Patel, 2007), a tool which also involves presentation, the approach selected in this thesis includes reporting tools that help the

investigator to easily assess data found in the selected Windows system generated artefacts.

The work undertaken in this research is similar to Carvey's (2007) Perl script, as the same mechanism is used to identify information in the artefacts. Data structures are extracted from a file, and field offsets extracted from an artefact are used as field descriptors of an artefact to extract information (as elaborated upon in Section 6.3) and not rely on Windows API.

### **6.3 Objective Evaluations Experiment of Forensic Analysis System**

This section describes the analysis performed by the prototype forensic analysis system on a selection of event log and swap files sourced from data collected as part of the disk studies carried out at the University of Glamorgan (Jones et al., 2006, 2008, 2009). The experimentation platform consisted of a PC (Pentium 4, with 4 GB memory) running under a Windows XP Professional environment (32 Bit, Version 2002) and during the experiment no other programs were running on the PC.

After retrieving and examining evidence data using the prototype tool, SAFTool, the results were verified by performing the same tasks with comparable forensic software. The experiment used a number of pieces of software for comparison: Event Viewer, which is a GUI manager for the Event logs by Microsoft; Carvey's Perl script (evtstats.pl and lsevt.pl), which is the script which parses the header of the Event logs file and determines the number of records that should exist (for the Event logs); and WinHex (Vyavhare, 2009), a hexadecimal editor to validate the results of the prototype tool (for the Swap files). The Event Viewer, evtstats.pl and lsevt.pl, and WinHex will be examined along with the prototype tool for evaluation purposes. The results of using these forensic tools over the same test data is compared to the SAFTool results. The tools should retrieve the same information when the same event logs file or swap file is analysed and presented.

Measuring the ability to retrieve information required visualising the information extracted from Windows system generated artefacts in the form of narrative constructs and also in graphical form. The comparison and accuracy of information retrieved by the tools was based on the number of records and the content recovered from the artefacts. Since our prototype tool, the SAFTool had not been pre-assessed for its extracting and visualising capabilities, we used the accuracy scores relating to the data extracted (number of records parsed) from each file to determine whether the tool was functioning as desired under normal conditions. Snapshots were collected of the tool's functioning capability.

## **6.4 Practical Experiments**

This section describes the practical experimentation. As discussed in Chapter 4, this took place over three experiments, each focused on determining whether the objectives (Section 1.3.3) had been met.

As discussed in Section 6.1, the first and second practical experiments assess the functionality of the tool. This assessment was to determine whether the prototype tool was functioning according to the design requirements.

- The first experiment examined the SAFTool to determine whether the application was functioning as desired under normal conditions. Screenshots were collected from this process.
- The second experiment examined the SAFTool as a whole against the nominated forensic software (Event Viewer, evtstats.pl, lsevt.pl and WinHex), to discover how the SAFTool compared to existing solutions (Section 6.4.1 until Section 6.4.5).
- In this experiment, the analysis involved determining the SAFTools ability to parse the event logs and swap files and display the results in a suitable format (Section 6.4.6).

- In this experiment, the variables examined in this experiment are the number of records parsed with processing time from the selected event logs and swap files (Section 6.4.7).
- The third experiment examined the SAFTTool to determine how well the system supported the analysis of Windows system generated artefacts by evaluating the functionalities and usefulness of the SAFTTool (Section 6.5).

To carry out the following experiments, a dataset was created that contained 10 event log files and 10 swap files (pagefile.sys). A list of the files used and associated information can be seen in Appendix G.

#### **6.4.1 Experiment on the Event Logs with the Event Viewer**

The following steps took place when running this experiment:

1. The dataset from Appendix G was inserted into the Windows directory of the test computer.
2. The Event Viewer application was invoked and set to view the events. Screenshots, number of records parsed and processing times were collected from this process.
3. The tool, as it is part of Windows, provides output via a Windows GUI. When data started to appear in the Event Viewer application, the Action menu was clicked when the mouse position was on the left pane and next the Action menu was clicked when the mouse position was on the right pane.

#### **Results**

A visual record of the output from the Microsoft event viewer is shown in Figure 6.1. The Event logs are presented to the user in a Windows Explorer like interface. Event Viewer tool has two different panes. The left pane shows the list of available log files, the right pane shows the selected log's contents. The left pane shows the list of available log files: Application, System and Security log, while the right pane shows a list of different event

entries, one line per entry. The menu options in Event Viewer are context sensitive, depending on where the focus of the mouse is at the time. The menu choices may change. In having two panes, and context sensitive menu option make the tool more difficult to use as users can be confused at what information they are looking at. Opening a saved log is not a straightforward process. There are four items that the Event Viewer has to be told: the File name, Files of Type, Log Type and Display Name. This means that opening a file is not possible or wrong information is displayed if the items given is not correct. In Figure 6.1, the Event Viewer is shown showing the Windows Event logs file in two panes with context sensitive Action menu for one Event log test subject. This Figure 6.1 shows the result from the third step of the experiment. Sometime when opening a log files that were saved or imaged from another system, the Event Viewer gives an error message that the file is corrupted.

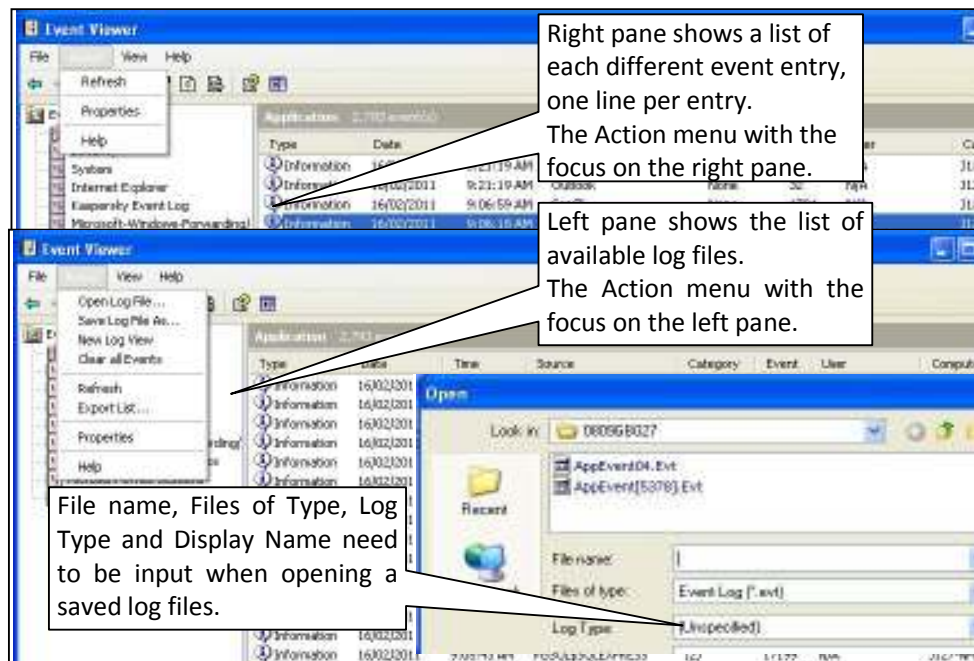


Figure 6.1: The Event Viewer Shows the Content of the Windows Event Logs

### 6.4.2 Experiment on the Event Logs with the `evtstats.pl` and `lsevt.pl`

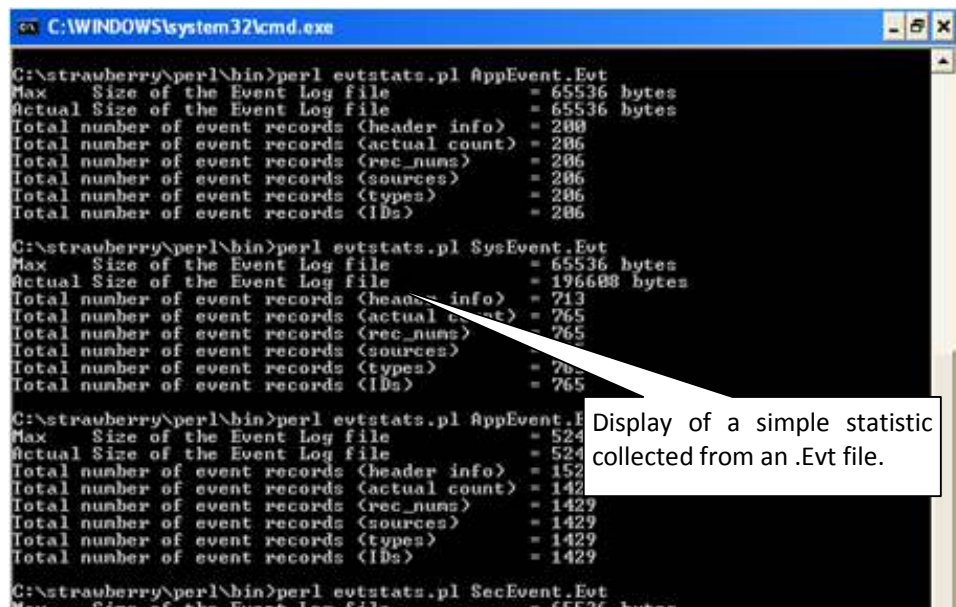
This experiment examined how `evtstats.pl` and `lsevt.pl` could view events in an event log file. The following steps took place when running this experiment:

1. The dataset listed in Appendix G was inserted into the Windows directory of the test system.
2. The `evtstats.pl` application was invoked and set to view the events.
3. The `lsevt.pl` application was invoked and set to view the events.
4. Screenshots, number of records parsed and processing times were collected from this process.
5. The event log files were processed and appeared in the Command Prompt window.

#### Results

The visual output can be seen in Figures 6.2 and 6.3. The Perl scripts of Carvey (2007) collected information from the Event log files and displayed simple statistics collected from the `.Evt` file shown in Figure 6.2 and displayed event records in simple listing format as illustrated in Figure 6.3. The mechanism operated like this: the script parsed the header of the Event log file and determined the number of records that should exist, then parsed through the contents of the Event log file and, using various tags from within each event record, performed an actual count of the number of records found.





```

C:\WINDOWS\system32\cmd.exe

C:\strawberry\perl\bin>perl evtstats.pl AppEvent.Evt
Max Size of the Event Log file = 65536 bytes
Actual Size of the Event Log file = 65536 bytes
Total number of event records (header info) = 200
Total number of event records (actual count) = 206
Total number of event records (rec_nums) = 206
Total number of event records (sources) = 206
Total number of event records (types) = 206
Total number of event records (IDs) = 206

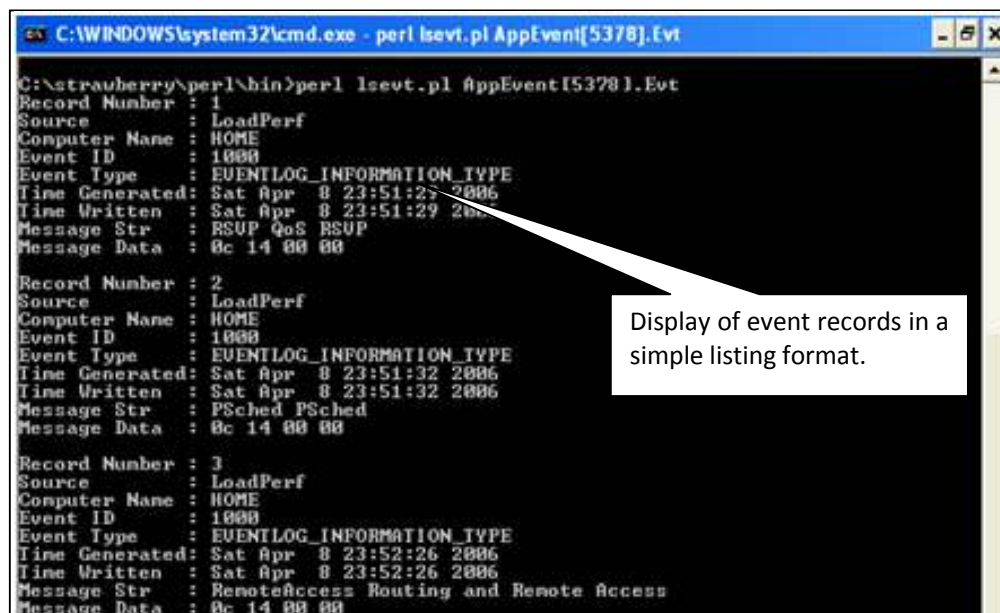
C:\strawberry\perl\bin>perl evtstats.pl SysEvent.Evt
Max Size of the Event Log file = 65536 bytes
Actual Size of the Event Log file = 196688 bytes
Total number of event records (header info) = 713
Total number of event records (actual count) = 765
Total number of event records (rec_nums) = 765
Total number of event records (sources) = 765
Total number of event records (types) = 765
Total number of event records (IDs) = 765

C:\strawberry\perl\bin>perl evtstats.pl AppEvent.Evt
Max Size of the Event Log file = 524
Actual Size of the Event Log file = 524
Total number of event records (header info) = 152
Total number of event records (actual count) = 142
Total number of event records (rec_nums) = 1429
Total number of event records (sources) = 1429
Total number of event records (types) = 1429
Total number of event records (IDs) = 1429

C:\strawberry\perl\bin>perl evtstats.pl SecEvent.Evt
Max Size of the Event Log file = 65536 bytes

```

Figure 6.2: The evtstats.pl Shows the Statistic from an Event Log File



```

C:\WINDOWS\system32\cmd.exe - perl lsevt.pl AppEvent[5378].Evt

C:\strawberry\perl\bin>perl lsevt.pl AppEvent[5378].Evt
Record Number : 1
Source : LoadPerf
Computer Name : HOME
Event ID : 1000
Event Type : EVENTLOG_INFORMATION_TYPE
Time Generated: Sat Apr 8 23:51:27 2006
Time Written : Sat Apr 8 23:51:29 2006
Message Str : RSUP QoS RSUP
Message Data : 0c 14 00 00

Record Number : 2
Source : LoadPerf
Computer Name : HOME
Event ID : 1000
Event Type : EVENTLOG_INFORMATION_TYPE
Time Generated: Sat Apr 8 23:51:32 2006
Time Written : Sat Apr 8 23:51:32 2006
Message Str : PSched PSched
Message Data : 0c 14 00 00

Record Number : 3
Source : LoadPerf
Computer Name : HOME
Event ID : 1000
Event Type : EVENTLOG_INFORMATION_TYPE
Time Generated: Sat Apr 8 23:52:26 2006
Time Written : Sat Apr 8 23:52:26 2006
Message Str : RemoteAccess Routing and Remote Access
Message Data : 0c 14 00 00

```

Figure 6.3: The lsevt.pl Shows the Event Records from an Event Log File

### 6.4.3 Experiment on the Event Logs with the SAFTool

This experiment examined (i) the built in extensible architecture of the SAFTool by testing the mechanism used to install a new plugin; and (ii) how the SAFTool examined and viewed events in an event log file. The following steps took place when running this experiment:

1. The dataset from Appendix G was inserted into the Windows directory of the test system.
2. The SAFTool was run. Initially no option of artefact names could be seen on the menu.
3. The Install Plugin was clicked; the process of installing the plugin was performed.
4. The Analyse Event Log was clicked; the process of analysing event logs was performed.
5. The Visualise Event Log was clicked; the process of visualising events for event logs was performed.
6. The Report Event Log was clicked; the process of reporting the event logs was performed. Screenshots, number of records parsed, and the processing times were collected during the experiment.

### Results

In Chapter 5, the extensible architecture, on which the prototype implementation was based, was outlined. The architecture enables a variety of system artefacts to be analysed by enabling other system objects and tools to be added easily; and it also caters for different data structures. In other words, a number of possible complex data structures within Windows can be easily analysed using the developed architecture. ‘Plug and Play’ features can be added easily, depending on what exactly is being analysed, how the analysis is to be shown, and what type of case is under investigation.

The Windows Event logs and Swap files were investigated to demonstrate that the proposed extensible architecture could extract information from the known data structure of Windows Event logs and the

not known data structure of Swap files. The architecture also allows features to be added to support visualisations and various forensic analysis capabilities. A concept of plugin is introduced to the architecture. A plugin can be developed to perform a specific function on the data. The plugin can perform searches on the data sets data and determine the most appropriate way to display the results in the interface. In order to examine the architecture's extensibility, the plug and play features were examined.

The visual output can be seen in Figure 6.4. The user interface of the SAFTool is dynamic. It is dynamic in such a way that it shows the implementation of the extensible architecture of the application. As illustrated in Figure 6.4, the prototype tool did not display the list of artefact name files for the user to choose to analyse (Analyse menu), further analyse (Visualise menu) and report (Report menu) before an artefact's plugin had been installed. Furthermore, the Analyse menu, Visualise menu and Report menu were disabled. Only the File menu was enable to be clicked for installing the plugin, to restart the application when a plugin was to be reinstalled, and exit the application. When a plugin was installed, the Analyse menu, Visualise menu and Report menu were enabled and displayed the artefact name files. Figure 6.5, Figure 6.6 and Figure 6.7 illustrated the Analyse menu; Visualise menu; and Report menu were enabled and displayed the artefact name files. This feature allows the core application to be extended easily without modifying its original code base or core application and enhances its functionality with new plugins.

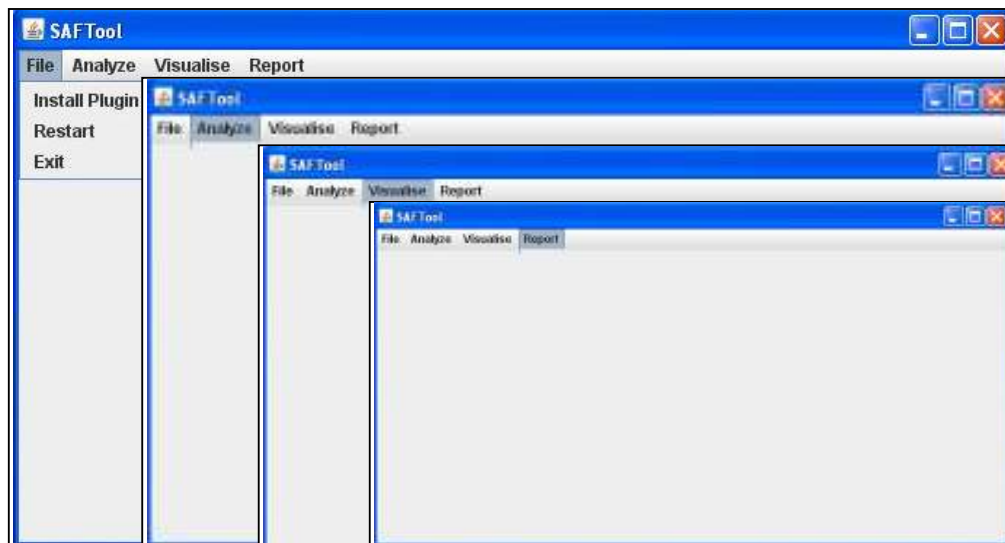


Figure 6.4: No Plugin Installed

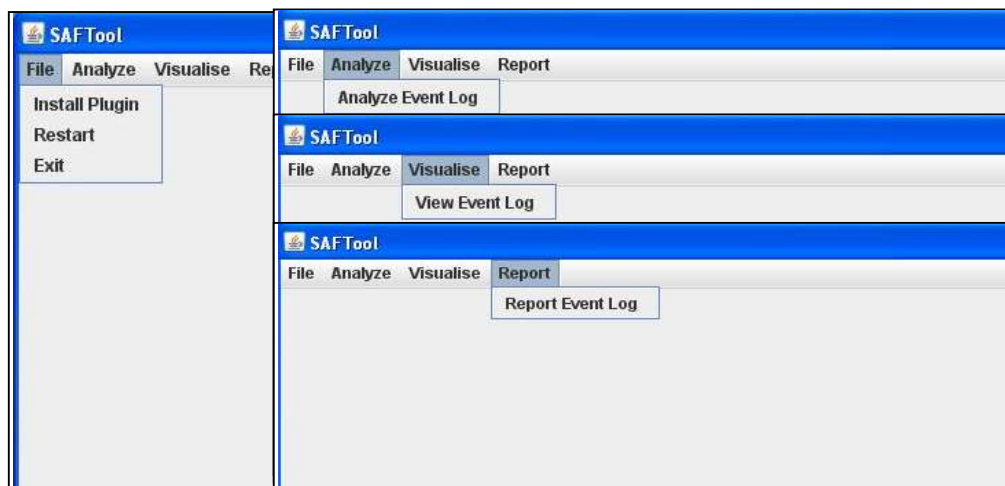


Figure 6.5: The SAFTool Displays Active Menu When the Event Logs Plugin Has Been Installed

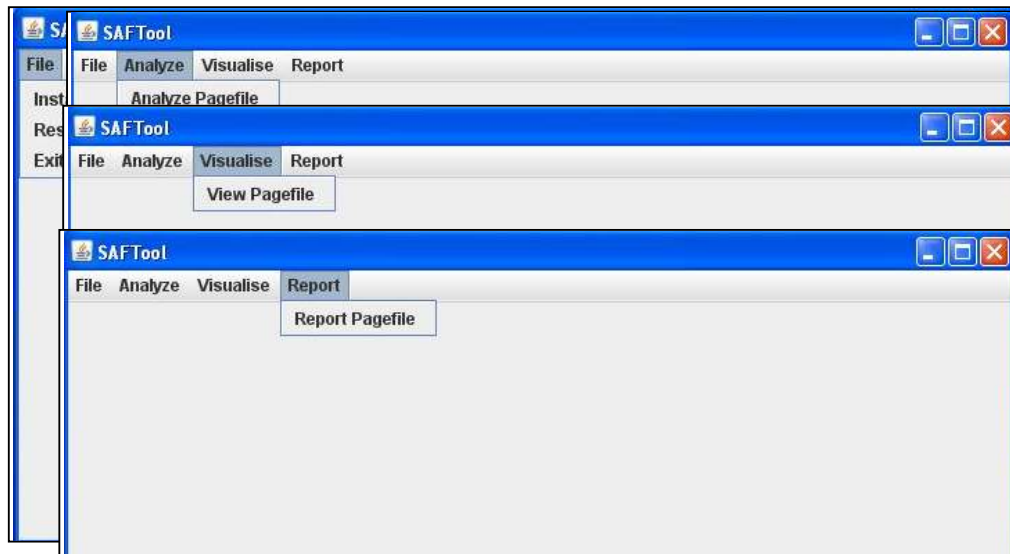


Figure 6.6: Th SAFTool Displays Active Menu When the Swap Files Plugin Has Been Installed

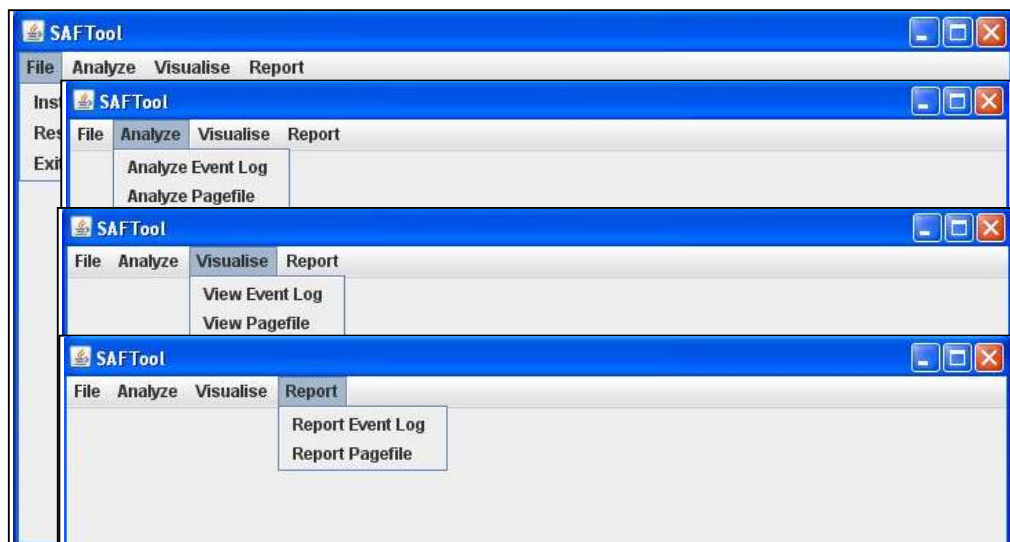


Figure 6.7: The SAFTool Displays Active Menu after the Event Logs and Swap Files Plugin Have Been Installed

The SAFTool allows the user to launch the application, and then select the specific system generated artefacts they wish to process. In this situation, the user can select the Event log as the artefact, and provide the location of the extracted files for analysis or allow the application to scan the computer for

any .Evt files available, thus the user can select the desired file to be examined. For example, in Figure 6.8, the SAFTool displays the list of event log files the user can choose to analyse.

The SAFTool shows the three main categories of event log (Application, Security and System) as a separate file. Once the files have been selected, the user can examine the data that has been processed by the prototype tool. The prototype tool extracts the following data from the files: Record number, Date and Time, Event ID, Event Type, Event Category, Source Name, Computer Name and Description. The user does not need prior knowledge of the internal data structure of the files, and the data is extracted.

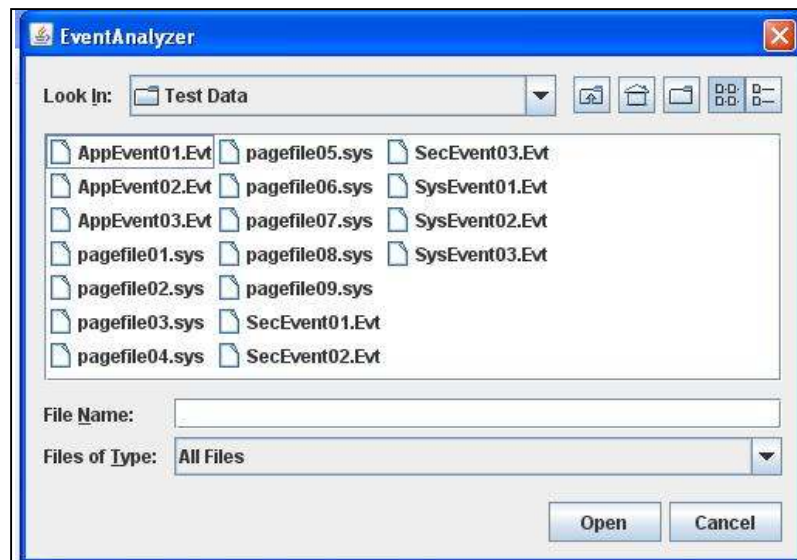


Figure 6.8: The SAFTool Display the Event Log Files the User Can Choose to Analyse

The logs in their native .Evt binary format provide the most flexibility in analysing them. The prototype tool parses through the information in those files in a manner that does not rely on the Windows API. This is important since this approach not only provides the user with the possibility of discovering hidden information or having loss of data if the file comes from other formats such as .txt or .csv. It also allows the user to perform analysis on platforms other than Windows; in other words investigators are not restricted to analysing Windows images on a Windows platform. The

visual output can be seen in Figure 6.9. As illustrated in Figure 6.9, the prototype tool displays the menu user can choose to analyse event logs and further display the time taken to process the event logs.

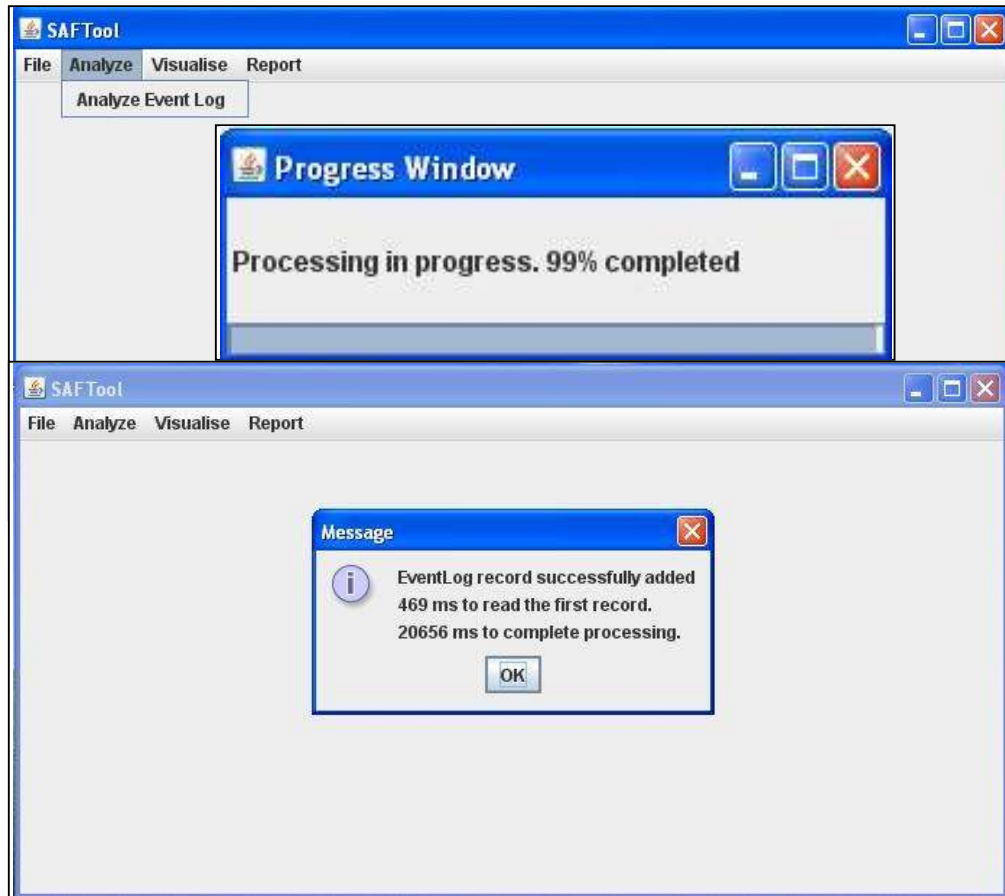


Figure 6.9: The SAFTool Displays the Menu Option, Progress Window and Processing Time Message Window for the Event Logs Analysis

The event records in the files are saved in the database for further analysis. Each category of event log has its own table in the database with its file name and timestamp for the table name. The purpose of choosing a file name and timestamp for the table name is to make it unique and simplify the retrieval for further analysis. The user does not need to know the table name and where the table is stored, the prototype tool automatically shows the artefact name and the user simply selects the desired artefact to be analysed. Further, the prototype tool offers the user (via a menu options) the opportunity

to visualise the event logs. As illustrated in Figure 6.10, the prototype tool is shown displaying the list of event file name the user can choose to view further.

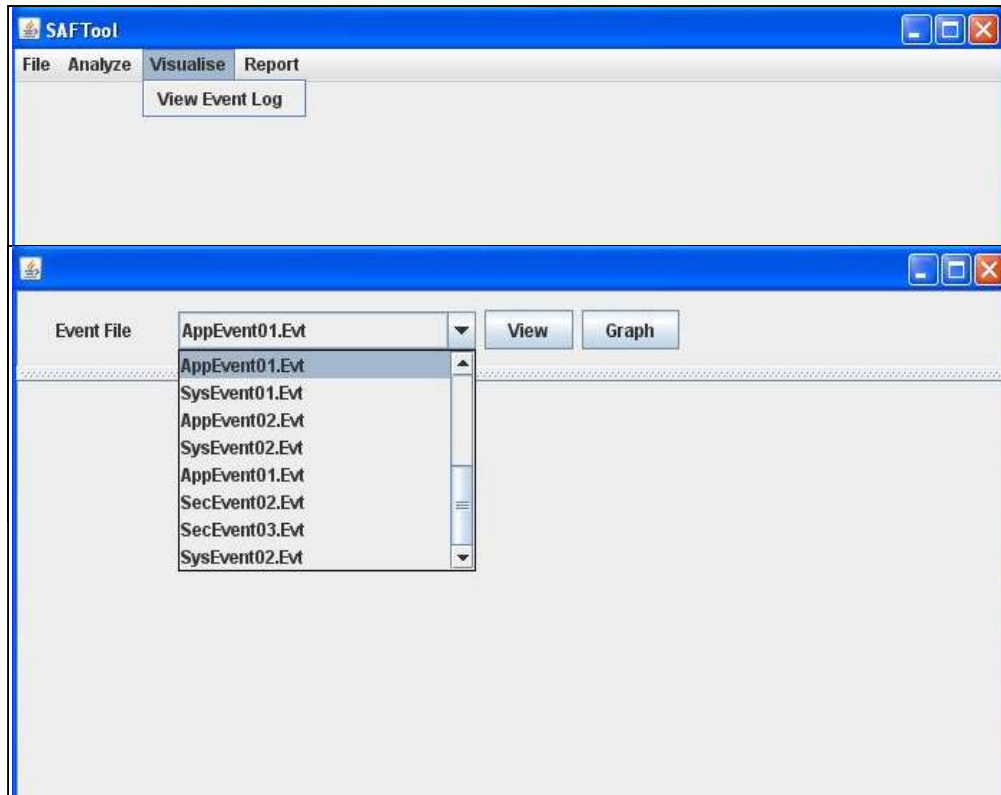
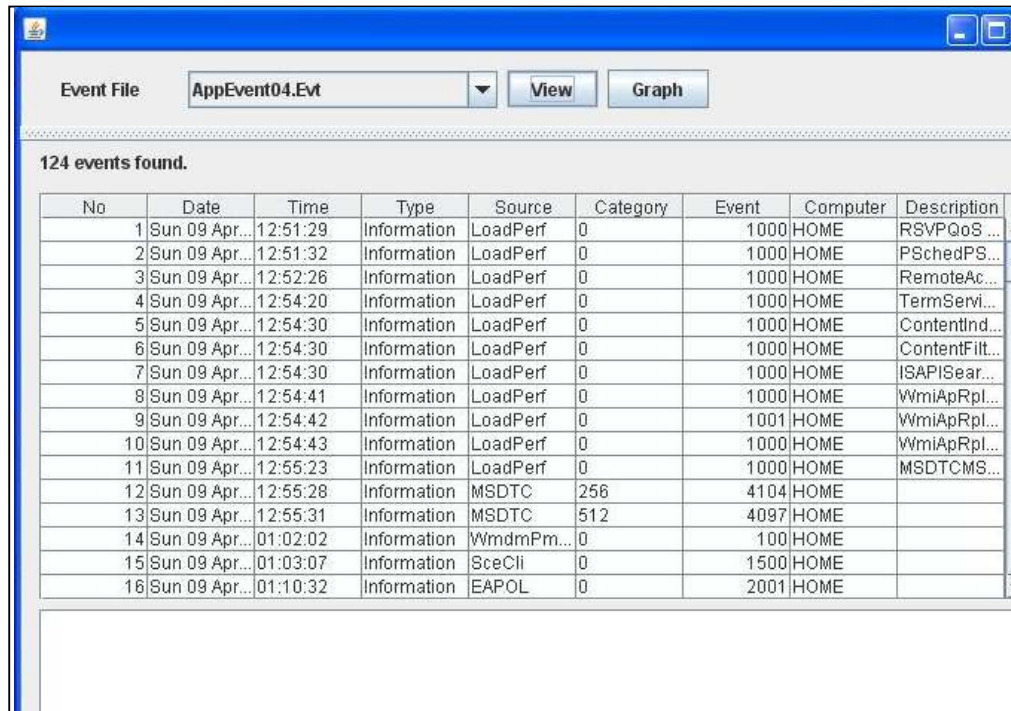


Figure 6.10: The SAFTool Displays the Menu Option and List of Event Logs Files User Can Choose to View Further

Further analysis is provided in the prototype tool for event logs. The user can select a specific event log and the visualisation format. There are two formats used in displaying the file contents of event logs. One format lists all the event records in an event log file, one line per entry. Figure 6.11 shows the SAFTool displaying the AppEvent04.Evt event records.





No	Date	Time	Type	Source	Category	Event	Computer	Description
1	Sun 09 Apr...	12:51:29	Information	LoadPerf	0	1000	HOME	RSVPQoS ...
2	Sun 09 Apr...	12:51:32	Information	LoadPerf	0	1000	HOME	PSchedPS...
3	Sun 09 Apr...	12:52:26	Information	LoadPerf	0	1000	HOME	RemoteAc...
4	Sun 09 Apr...	12:54:20	Information	LoadPerf	0	1000	HOME	TermServi...
5	Sun 09 Apr...	12:54:30	Information	LoadPerf	0	1000	HOME	ContentInd...
6	Sun 09 Apr...	12:54:30	Information	LoadPerf	0	1000	HOME	ContentFilt...
7	Sun 09 Apr...	12:54:30	Information	LoadPerf	0	1000	HOME	ISAPISear...
8	Sun 09 Apr...	12:54:41	Information	LoadPerf	0	1000	HOME	WmiApRpl...
9	Sun 09 Apr...	12:54:42	Information	LoadPerf	0	1001	HOME	WmiApRpl...
10	Sun 09 Apr...	12:54:43	Information	LoadPerf	0	1000	HOME	WmiApRpl...
11	Sun 09 Apr...	12:55:23	Information	LoadPerf	0	1000	HOME	MSDTCMS...
12	Sun 09 Apr...	12:55:28	Information	MSDTC	256	4104	HOME	
13	Sun 09 Apr...	12:55:31	Information	MSDTC	512	4097	HOME	
14	Sun 09 Apr...	01:02:02	Information	WmdmPm...	0	100	HOME	
15	Sun 09 Apr...	01:03:07	Information	SceCli	0	1500	HOME	
16	Sun 09 Apr...	01:10:32	Information	EAPOL	0	2001	HOME	

Figure 6.11: The SAFTool Displaying the AppEvent04.Evt Event Records

The second format displays the events information in graphical form and a timeline of event records. The graph shows the number of types of event which will entries relating to Information, Warning and Error for Application and System logs, and Success or Failure for the Security log. According to Anson and Bunting (2007), “*some information, warning or error events may be of evidentiary value, the presence of a warning or an error by itself is not indicative of an attack*”. The visual output can be seen in Figure 6.12. This Figure 6.12 shows the graph displaying the number of entries of two types in this particular log, information and warning events. The timeline of event log records is the graphical representation of events in chronological order. The timeline portray events as they occurred and tells what happened as it happened (see Figure 6.13). According to Anson and Bunting (2007), “*timelines are an excellent means of conveying technical facts in a way that makes understanding much easier*”.

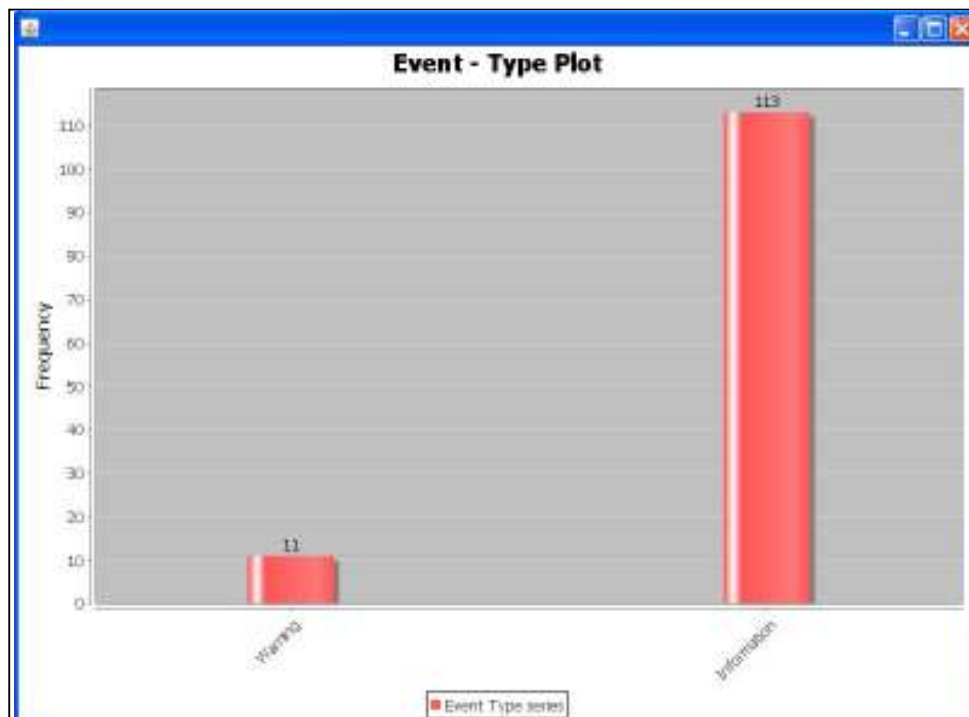


Figure 6.12: The Types of Events for the Event Logs

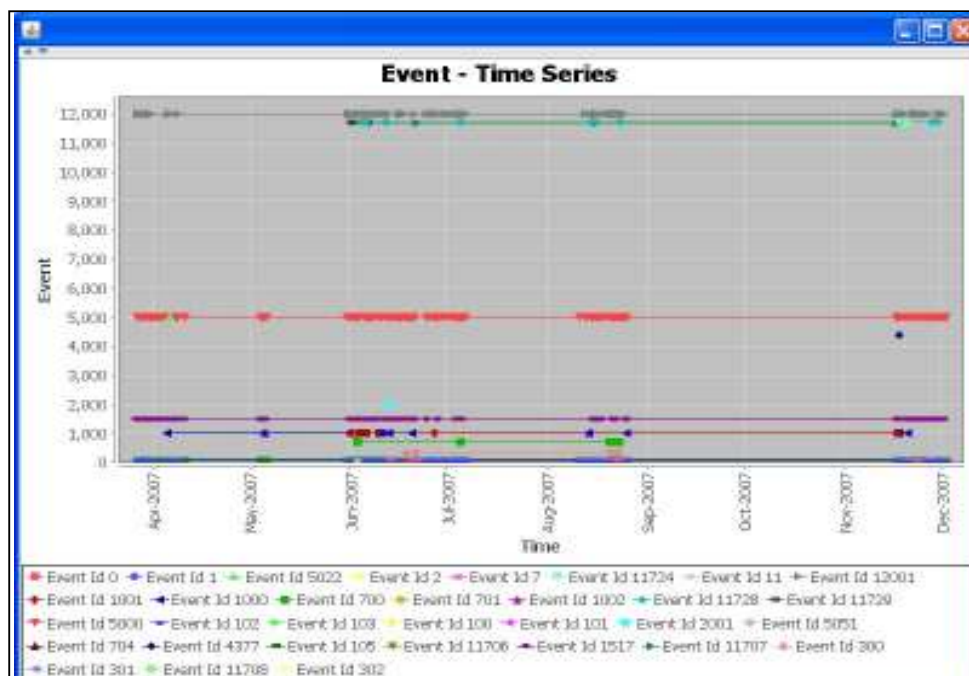


Figure 6.13: The Timeline of Event Logs Records

Once an investigator has located the information they require that information needs to be included into a report. As illustrated in 6.14, the prototype tool provides a feature to generate a report. Upon clicking the generate report option, the Microsoft document will pop up with information about the file name analysed, the date the file was created, the file hash value and the number of records existed in the file. At this point, a report has been created and been saved (see Figure 6.15 below).



Figure 6.14: The SAFTool Allows the User to Generate A Report for the Event Logs Analysis

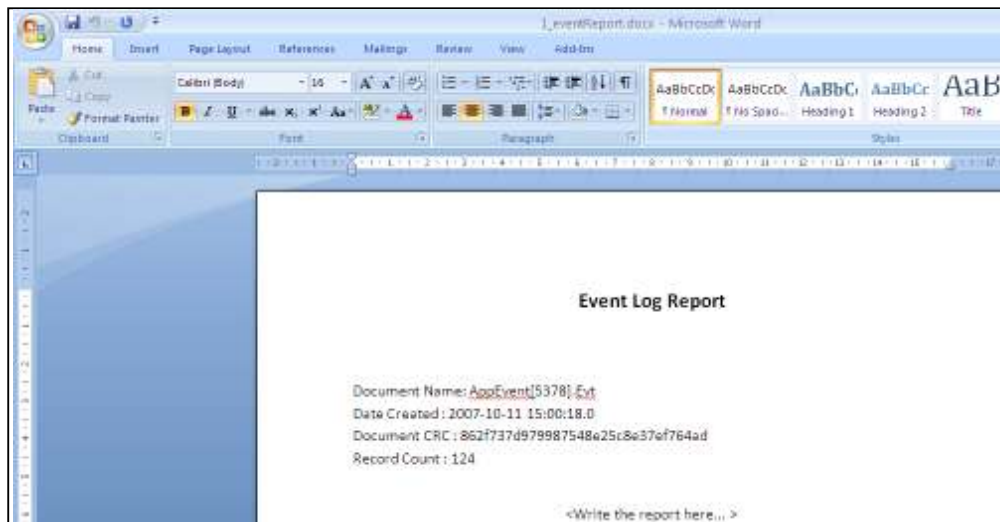


Figure 6.15: The Report Generated for the Event Logs Analysis

#### 6.4.4 Swap Files Analysis Using WinHex

The following steps took place when running this experiment:

1. The file listed in Appendix G was inserted into the Windows directory of the test system.
2. The WinHex application was invoked and set to view the swap files' content. Screenshots, total size of data and processing times were collected from this process.

#### Results

As discussed in Chapter 3, WinHex is the tool provided by X-Ways Software Technology AG and is capable of performing a number of operations including displaying the content of each file type, including swap files.

The user can select to open the swap files from the File menu of the WinHex window. The data contained within the swap files are presented to the user in GUI in a hexadecimal display and an ASCII display window (although there are options to display in a full width window).

WinHex provides the user with information about the logical size of the file (size without slack) or physical size of a directory, physical file size and valid data length (for files stored in an NTFS file system) in the Info Pane. The visual output can be seen in Figure 6.16 which illustrates the hexadecimal data display, the ASCII text display, and the information window.

The WinHex Search Menu provides a find text command to search for a specified string of ASCII characters in the current file. The visual output from WinHex can be seen in Figure 6.17.

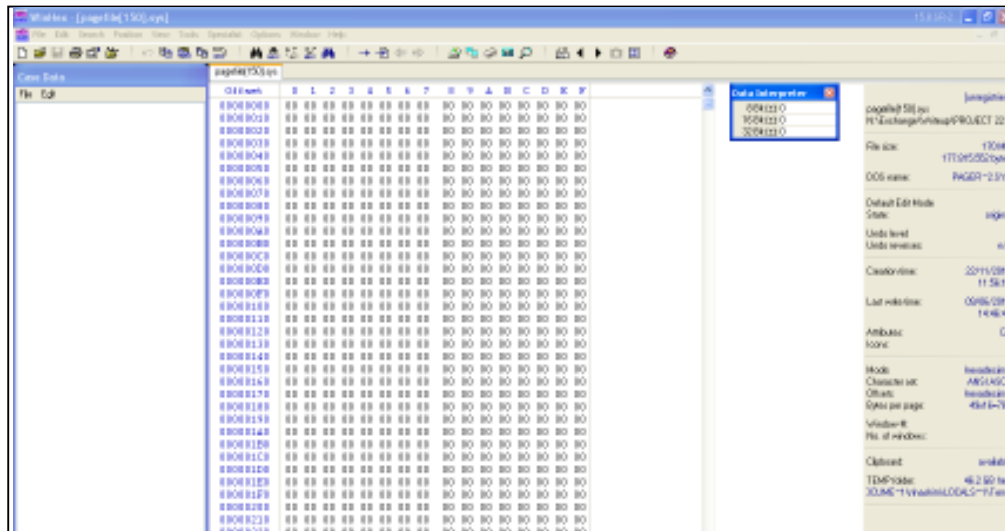


Figure 6.16: The Hexadecimal Data Display, ASCII Text Display And Info Pane for WinHex

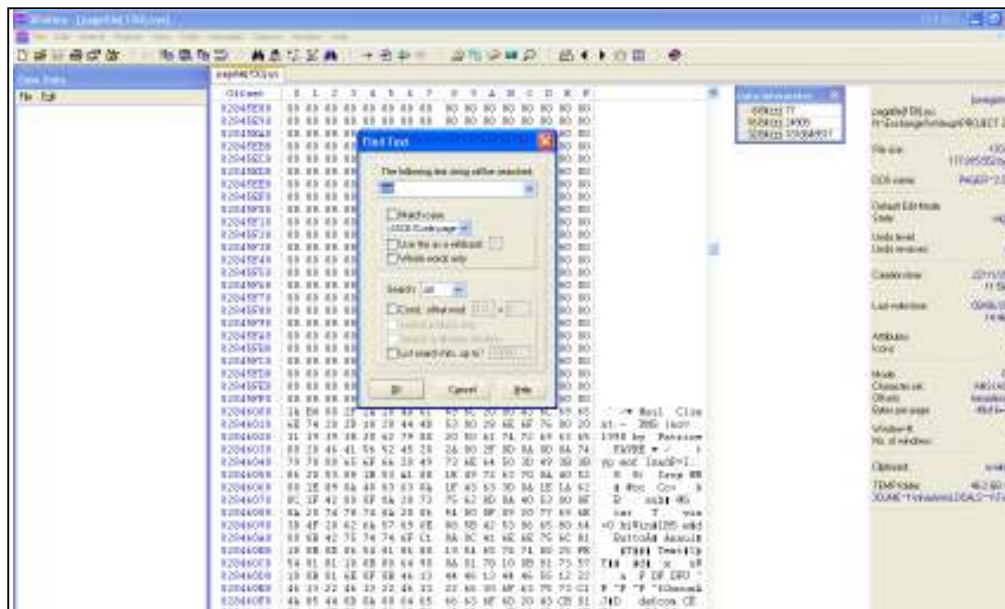


Figure 6.17: The Find Text Function for WinHex

### 6.4.5 Swap Files Analysis Using the SAFTool

This experiment investigated how the SAFTool can examine and views the contents of a swap file. The following steps took place when running this experiment:

1. The dataset from Appendix G was inserted into the Windows directory of the test computer.
2. The SAFTool was run.
3. The Analyse Pagefile menu was clicked; the process of analysing the swap file was performed.
4. The Visualise Pagefile menu was clicked; the process of visualising the swap file contents was performed.
5. The Report Pagefile menu was clicked; the process of reporting the examining of the swap file was performed. Screenshots, total size of data and processing times were collected from this process and are detailed below.

#### Results

The prototype tool allows the user to launch the application then select the system generated artefacts. In this situation, the user can select the swap files as the artefact, and provide the location of the extracted files for analysis or allow the application to scan the computer for any `.sys` files available, thus the user can select the desired file to be examined. The visual output can be seen in Figure 6.18 which illustrates the prototype tool displaying the list of swap files the user can choose to analyse.

Once the files have been selected, the user can examine the data depicted by the prototype tool. The user does not need prior knowledge of the internal data structure of the files.

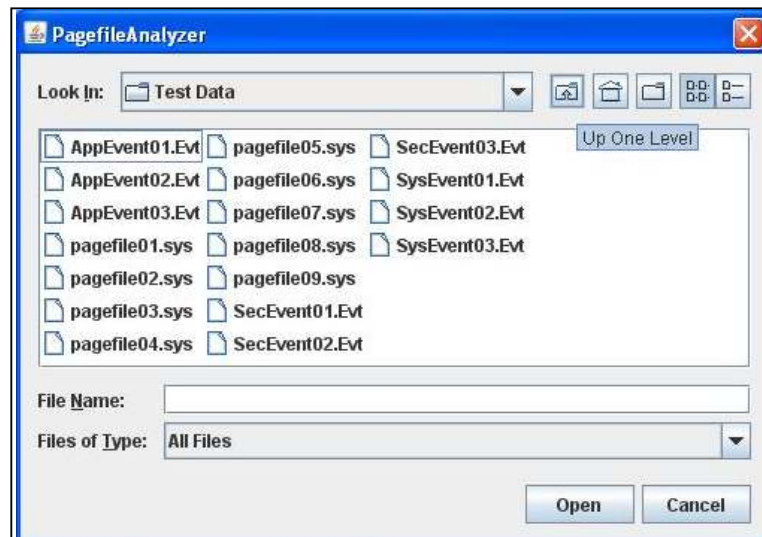


Figure 6.18: The SAFTool Displays the List of Swap Files the User Can Choose to Analyse

The swap files in their native binary format provide the most flexibility in analysing them. Collecting the swap file on a live system is cumbersome since the Windows operating system has complete control and protection of it. The prototype tool parses through the information in those files in a manner that translates a stream of bytes into a usable file structure of an artefact and recovers the content of the file. The visual output from the SAFTool can be seen in Figure 6.19. The prototype tool displays the menu option, the progress window, and the processing time for the swap files' analysis.



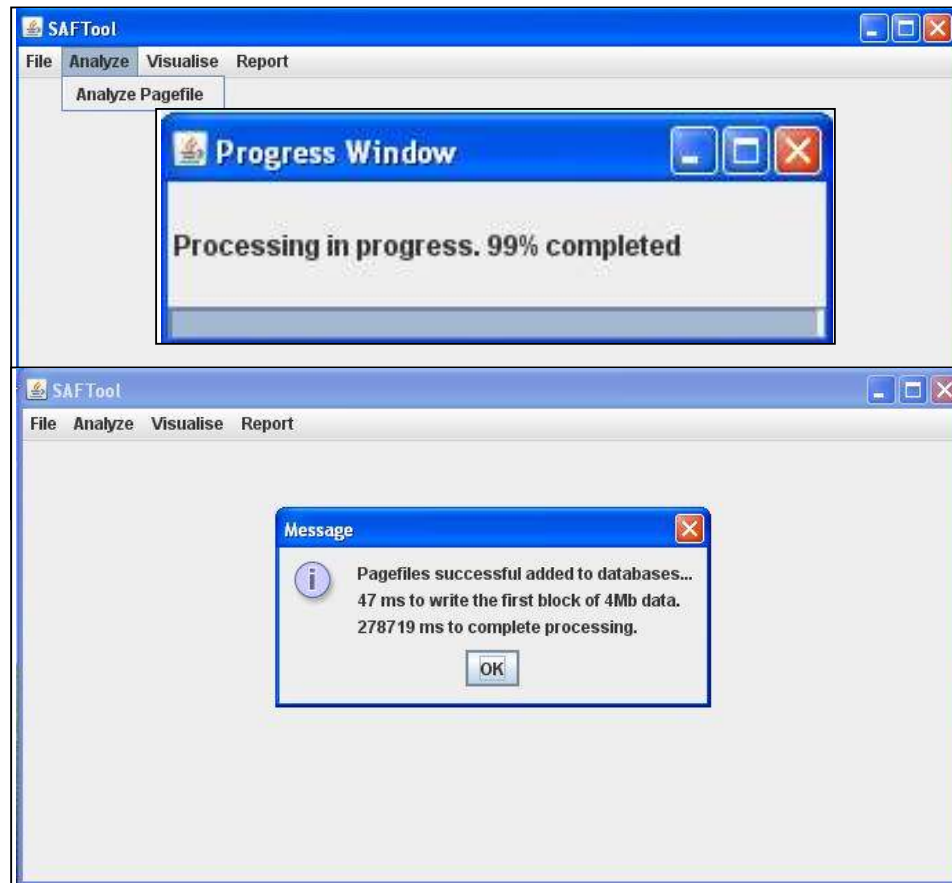


Figure 6.19: The SAFTool Displays the Menu Option, Progress Window and Processing Time Message Window for the Swap Files Analysis

The event records in the files are saved to the database for further analysis. Each category of event log has its own table in the database with its file name and timestamp for the table name. The purpose of choosing a file name and timestamp for the table name is to make it unique and retrievable for further analysis. The user does not need to know the table name and where the table is stored, since the prototype tool automatically shows the artefact name and the user simply selects the desired artefact to be analysed. Further, the prototype tool offers the user a menu to visualise the swap files as shown in Figure 6.20. This figure shows the prototype tool displaying the list of swap file names the user can choose to further analyse.



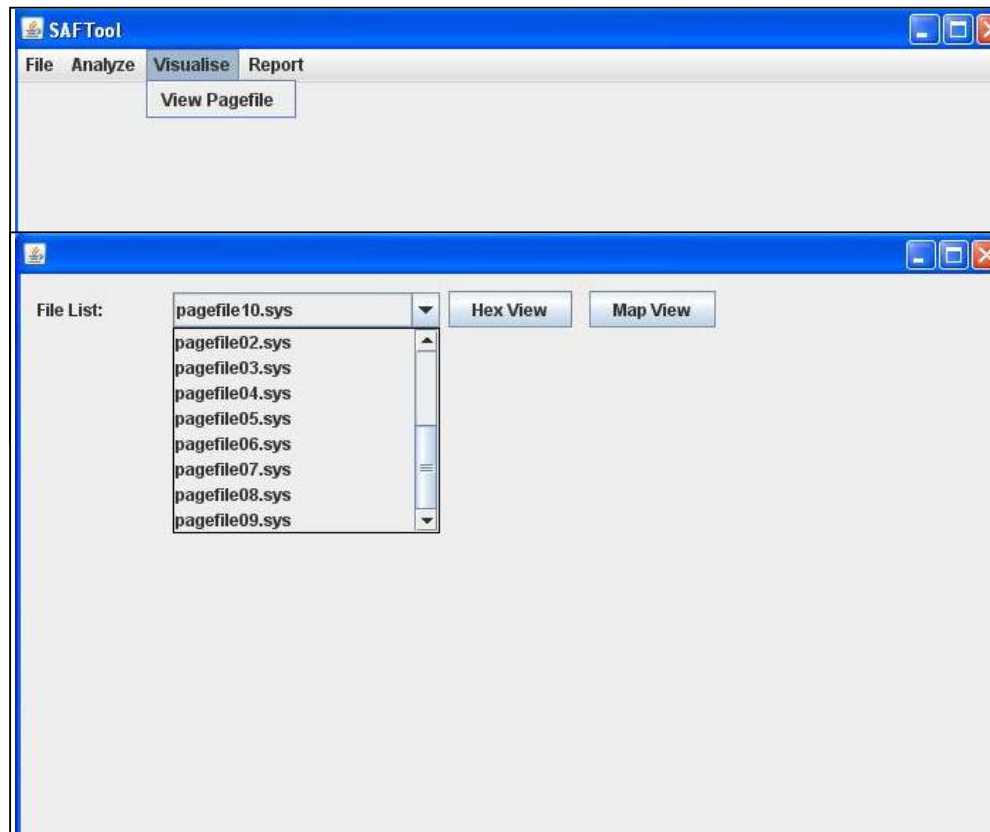


Figure 6.20: The SAFTool Displays the Menu Option and List of Swap File Names the User Can Choose to Visualise the Swap Files

The prototype tool supports the investigator by enabling the selection of specific swap file and a visualisation format for the file. Two possible visualisation formats have been included in the prototype tool. Firstly, the hexadecimal data display and ASCII text display format to display the detailed contents of each swap file. In Figure 6.21, output from the prototype tool displays the hexadecimal data display and ASCII text display format for the contents of swap files.

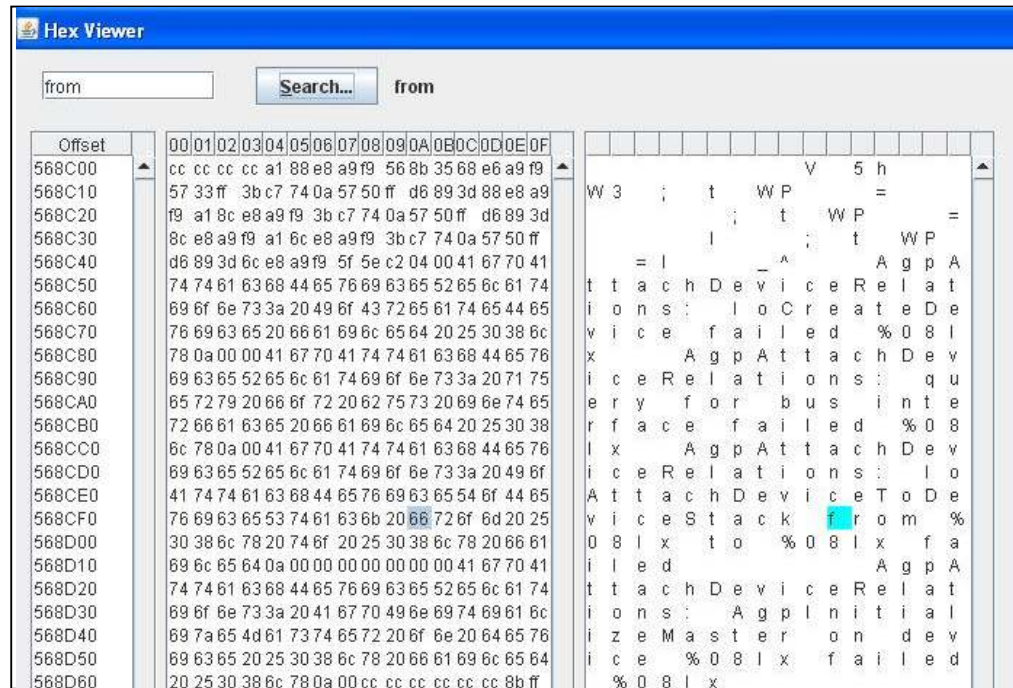


Figure 6.21: The Hexadecimal Data Display and ASCII Text Display Format for Displaying the Contents of Swap Files

The second visualisation included in the prototype is the block view format which displays the file's content. This consists of series of square blocks colour coded to indicate the density (the percentage of the ASCII character) of swap file's content. Each 4MB file's content is colour coded according to a predetermined scheme and is controlled by the percentage of the ASCII character in it. This is used due to, when the memory in use exceeds the amount of RAM available, the operating system will move pages (4KB pieces) of virtual address spaces to the swap file. 4MB is used instead of 4KB due to the size of swap file used which average was 555673KB. The block view format which displays the file's content is implemented as this visualisation technique (nonhierarchical visualisation technique defined by Teerlink and Erbacher (2006)), can aid the investigator to direct their search to suspicious area of the swap file. If the block's content contrasts greatly with an other block, it is easily spotted because it stands out against a sea of different coloured blocks. This method is used due to the fact that it is common that pages in swap files

are blank or contain less ASCII content. The visual output can be seen in Figure 6.22. As illustrates in Figure 6.22, the block view displays a swap file in which the density of the block is controlled by the intensity within the 4MB data of ASCII characters.

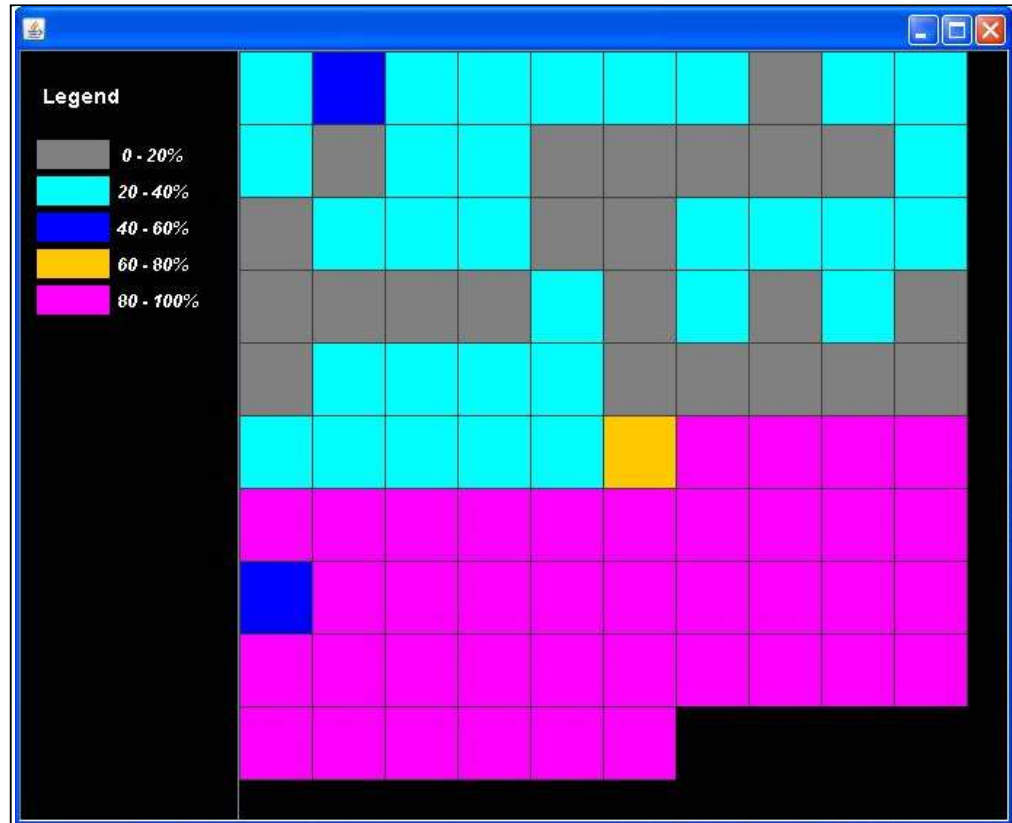


Figure 6.22: The Square Block Diagram for the Density of Swap Files

The prototype tool basic information need to be exported to a report. The visual output can be seen in Figure 6.23. The prototype tool provides a limited feature to generate a report. Upon clicking the generate report option, a Microsoft word document is created containing information about the file name analysed, the date the file created, the file hash value and number of records that exist in the file. At this point, a report has been created and have been saved, which is shown in Figure 6.24 below.

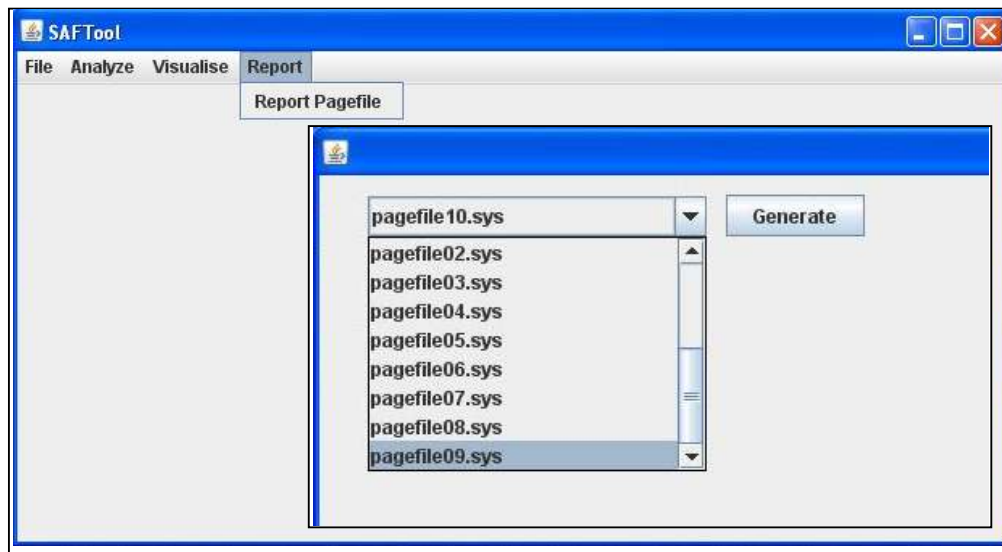


Figure 6.23: The SAFTool Enables the User to Generate A Report for the Swap Files Analysis

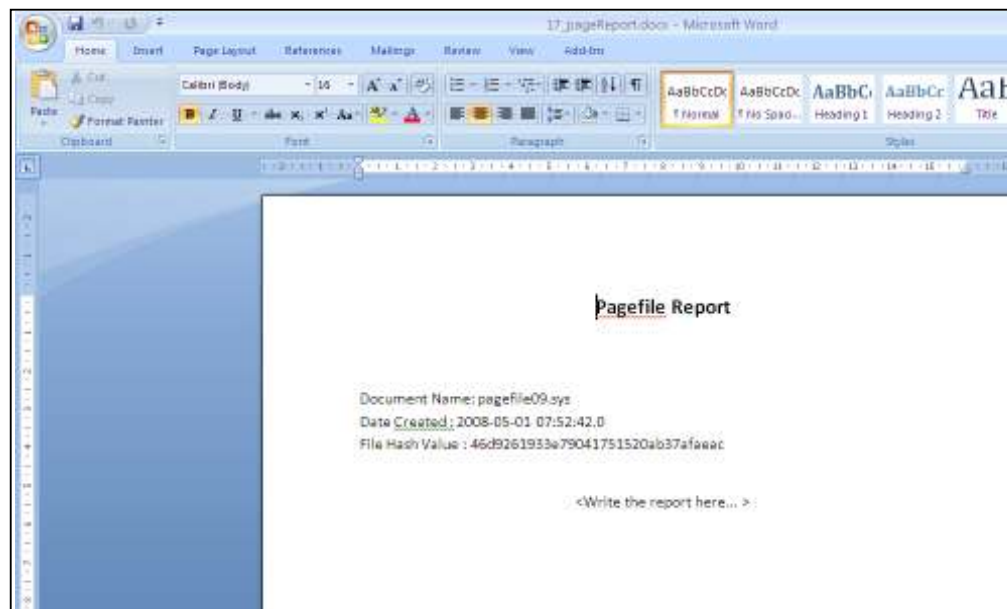


Figure 6.24: The Report Generated for the Swap Files.

### 6.4.6 Analysis of the Results

The following table (Table 6.6) below shows the results of the desired functions comparison. Overall, all the representatives were able to parse the test dataset and display the results accordingly. Table 6.6 shows, the SAFTool is able to parse system artefacts files and visualise the results in the forms of narrative constructs and also in graphical form, that is the tables, graph, timeline, hexadecimal data display, ASCII text display and block view.

Table 6.6: Ability to Parse the Dataset and Display the Results

Function	Event Viewer	evtstats.pl / lsevt.pl	SAFTool
Parse Event Logs	Yes. Windows API is used. Reports the saved event logs file is corrupted when trying to open it.	Yes. Windows API is not used.	Yes. Windows API is not used.
Format for Displaying Results	Two panes. One line per entry.	Simple listing format.	Narrative construct and graphical form, i.e. Tables, graph and time lines.

Function	WinHex	SAFTool
Parse Swap Files	Yes	Yes
Format for Displaying Results	Hexadecimal data display and ASCII text display.	Hexadecimal data display, ASCII text display and Block Diagram.

### 6.4.7 A Comparison of the Results for the Event Logs and Swap Files Experiments

Measuring the accuracy of this tools ability to retrieve information required assessment of the number of records in the artefacts with known and not known data structures, and evaluation of their visualisation in the form of a narrative construct and graph. For this evaluation, the experiments were performed on a number of artefact files of varying sizes that were extracted from different hard drive images (dataset from Appendix G) and using the tools mentioned earlier in Section 6.2.

The results of applying the various tools to the test files listed in Appendix G are summarised for discussion in Table 6.7. This table indicates the number of records that are processed from individual event logs (one line per log file). Ultimately, the SAFTTool was able to parse the files and locate a greater or equal number of records than the `evtsstat.pl`. The Event Viewer was not able to parse the files. When using Event Viewer as a tool to open and view the event records of a saved log file, viewing for the event records is impossible. This can be accounted for because the viewer relies on the eventlog service API. The file would be reported as corrupted because of the out of synch fields and the file status byte in both the header and the floating footer (Anson and Bunting, 2007).

Referring to the data shown in Table 6.7, there were two event log files examined by `evtstats.pl` and SAFTTool that generated different results, which were the `SysEvent01.Evt` and `AppEvent02.Evt` from the dataset in Appendix G. Both were obtained from parsing the Event log files in binary mode and bypassing the Windows API all together. According to Schuster (2007), such variation in results is likely due to a special condition regarding the Event log record, in that a record is written to the end of the `.Evt` file but wraps around the beginning of the file, resulting in part of the event record following the header.

Table 6.7: Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTool

Test Data File Name	Size (KB)	No. of Records		
		Event Viewer	evtstats.pl	SAFTool
AppEvent01.Evt	512	X	2317	2317
SecEvent01.Evt	64	X	no records found	no records found
SysEvent01.Evt	512	X	1364	1491
AppEvent02.Evt	512	X	1909	1911
SecEvent02.Evt	512	X	1949	1949
SysEvent02.Evt	512	X	2599	2599
AppEvent03.Evt	64	X	206	206
SecEvent03.Evt	64	X	260	260
SysEvent03.Evt	192	X	765	765
X – unable to complete the operation. The event log file is reported as corrupted.				

Figure 6.25 illustrates the total event records located for the evtstats.pl script and the prototype SAFTool.

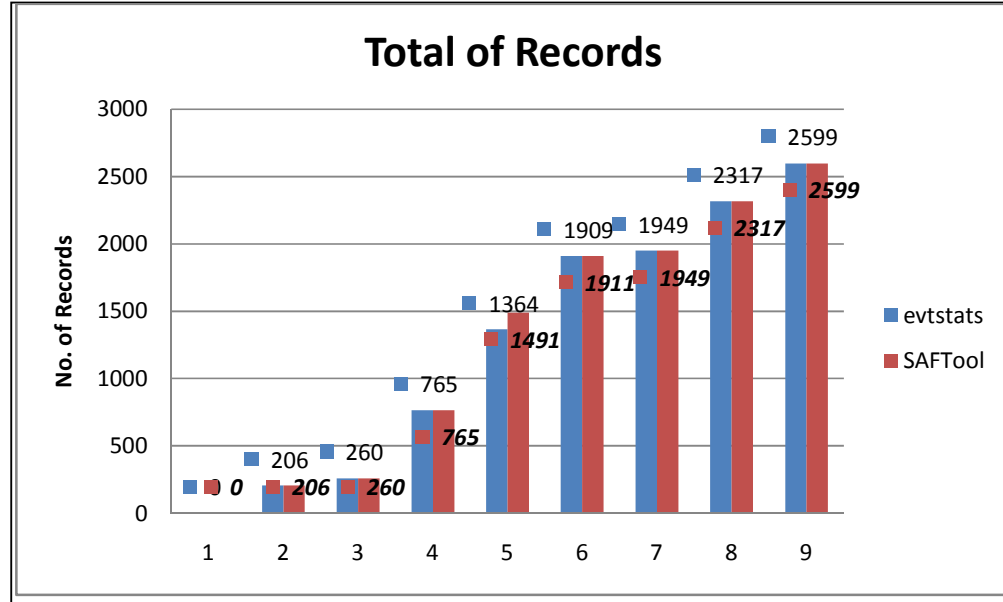


Figure 6.25: Total Event Records Located for the evtstats.pl Script and SAFTool

Data in Table 6.8 shows the processing times for the requests submitted to the forensic analysis system during the processing of the event logs. In the context of this chapter, processing time is the duration in time (measured in milliseconds (ms)) of the forensic analysis system to generate an analysis display upon receiving an analysis request. According to Zhu (2007), *“the common measure of efficiency is the time taken to complete a task and baseline task completion times should be recorded for non-visual displays and used as a reference”*.

Table 6.8: Processing Time for Event Viewer, lsevt.pl and SAFTool

Test Data File Name	Size (KB)	Event Viewer Processing Time (ms)	lsevt.pl		SAFTool	
			No. of Records	Processing Time (ms)	No. of Records	Processing Time (ms)
AppEvent01.Evt	512	X	2317	2172	2317	14813
SecEvent01.Evt	64	X	no records found	no records found	no records found	no records found
SysEvent01.Evt	512	X	1364	1641	1491	8379
AppEvent02.Evt	512	X	1909	1719	1911	10427
SecEvent02.Evt	512	X	1949	1852	1949	13032
SysEvent02.Evt	512	X	2599	2235	2599	16595
AppEvent03.Evt	64	X	206	719	206	1392
SecEvent03.Evt	64	X	260	859	260	1422
SysEvent03.Evt	192	X	765	953	765	5658
X – unable to complete the operation. The event log file is reported as corrupted.						

Table 6.8 shows the time required to process the event logs using lsevt.pl and SAFTool. It should be noted that the lsevt.pl tool is used in this comparison due to the fact that evtstats.pl only displays simple statistic mean while lsevt.pl displays event records in a listing format. The graph in Figure 6.26 is generated based on the parses through the contents of the Event log file and display event records. The results show more time is required by the SAFTool to parse through the Event log file (with a greater or equal number of records when compare to lsevt.pl) and display the event records. The total processing time would be in milliseconds and the practical impact is that extra



in average 8965 milliseconds is taken. This would suggest that, due to the visualisation of the analysis in a graphical form, the processing time is in average 8965 milliseconds; and the Carvey program (lsevt.pl) shows numbers of record in a simple listing format. Both illustrate almost the same shapes of graph, i.e. processing time increases with the file sizes. Note the fact that the Event Viewer supplied with the operating system was unable to deal with a saved event logs, the experiment with the Event Viewer could not be conducted.

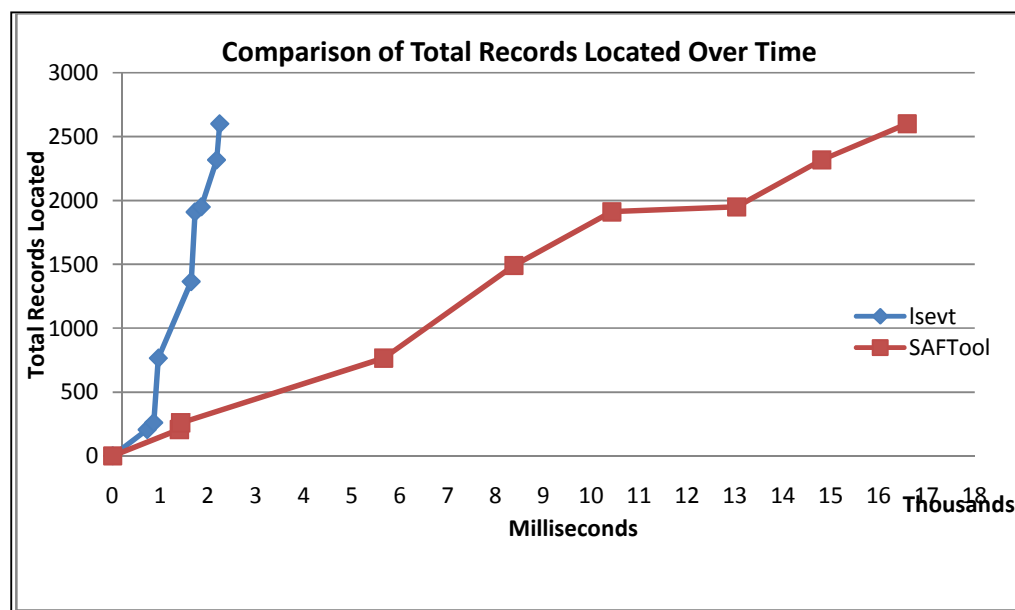


Figure 6.26: Graph Showing the Number of Records Located With Processing Time

Data in Table 6.9 shows total size of data parsed for WinHex and the SAFTTool. As can be seen, the same results were recorded for each tool. A graph (see Figure 6.27) was subsequently drawn comparing the total size of data parsed by each tool. It confirmed the result in Table 6.9.

Table 6.9: Total Size of Data for WinHex and the SAFTTool

Test Data File Name	Size (KB)	Total Size of Data (bytes)	
		WinHex	SAFTTool
pagefile01.sys	1,506,576	1,598,029,824	1,598,029,824
pagefile02.sys	540,672	553,648,128	553,648,128
pagefile03.sys	786,432	805,306,368	805,306,368
pagefile04.sys	774,144	792,723,456	792,723,456
pagefile05.sys	117,760	120,586,240	120,586,240
pagefile06.sys	512,000	524,288,000	524,288,000
pagefile07.sys	173,648	177,815,552	177,815,552
pagefile08.sys	196,608	201,326,592	201,326,592
pagefile09.sys	393,216	402,653,184	402,653,184

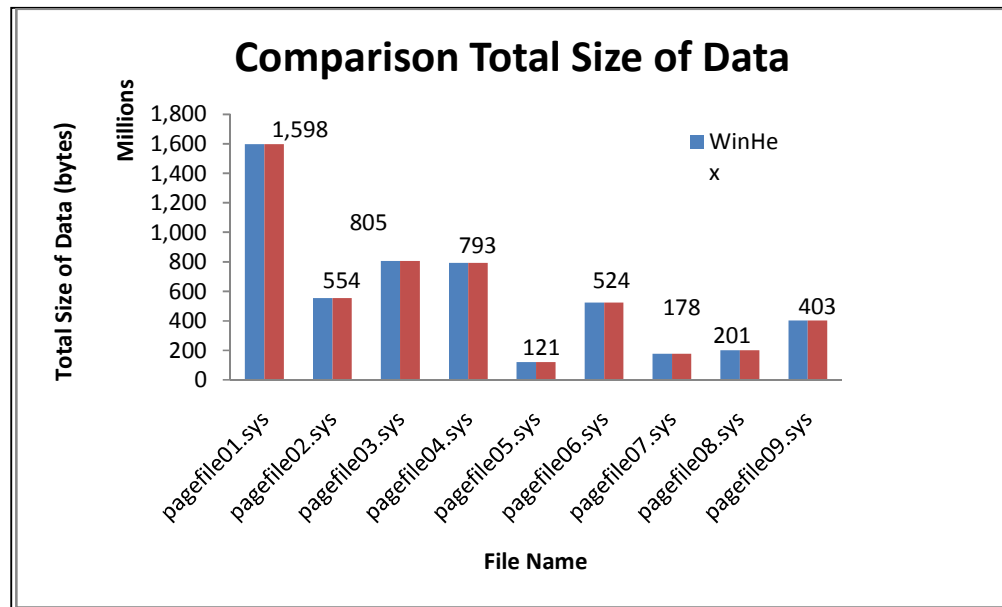


Figure 6.27: Comparison of Total Size of Data Shown by the WinHex and SAFTTool

Table 6.10 shows the processing times for WinHex and the SAFTTool which was subsequently displayed in graph form, Figure 6.28 and Figure 6.29 respectively.

In the context of this chapter, processing time recorded in milliseconds (ms) in term of the time taken by the forensic analysis system to generate an analysis display. It took a longer generate hexadecimal data display or an ASCII text display for the contents of each swap file when the SAFTTool was employed. The additional time was required as a result of the need for the SAFTTool to find the data structure and to identify the structure of each record in a binary data file without using a predefined template, WinHex was slightly faster due to the uses of a pre-defined template to examine a swap file and has the capability to move freely forwards and backwards within the data (X-Ways Software Technology AG, 2010).

Table 6.10: Processing Time for WinHex and SAFTTool

Test Data File Name	Size (KB)	Processing Time (ms)	
		WinHex	SAFTTool
pagefile01.sys	1,506,576	531	575,484
pagefile02.sys	540,672	469	235,078
pagefile03.sys	786,432	516	292,579
pagefile04.sys	774,144	500	266,562
pagefile05.sys	117,760	297	38,343
pagefile06.sys	512,000	437	137,063
pagefile07.sys	173,648	390	50,297
pagefile08.sys	196,608	406	55,438
pagefile09.sys	393,216	421	98,765

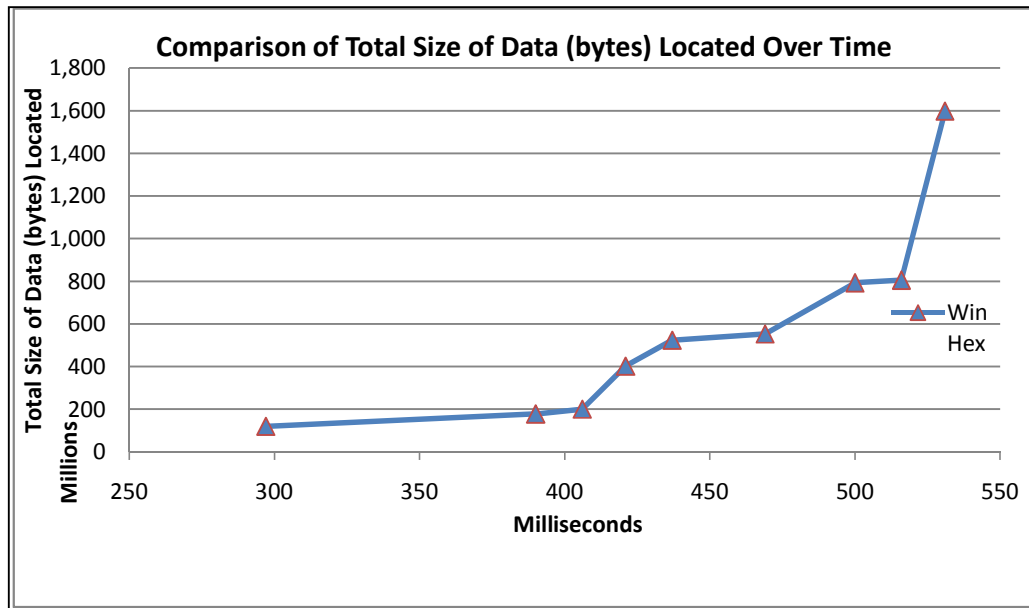


Figure 6.28: Total Size of Data Located Over Time for WinHex

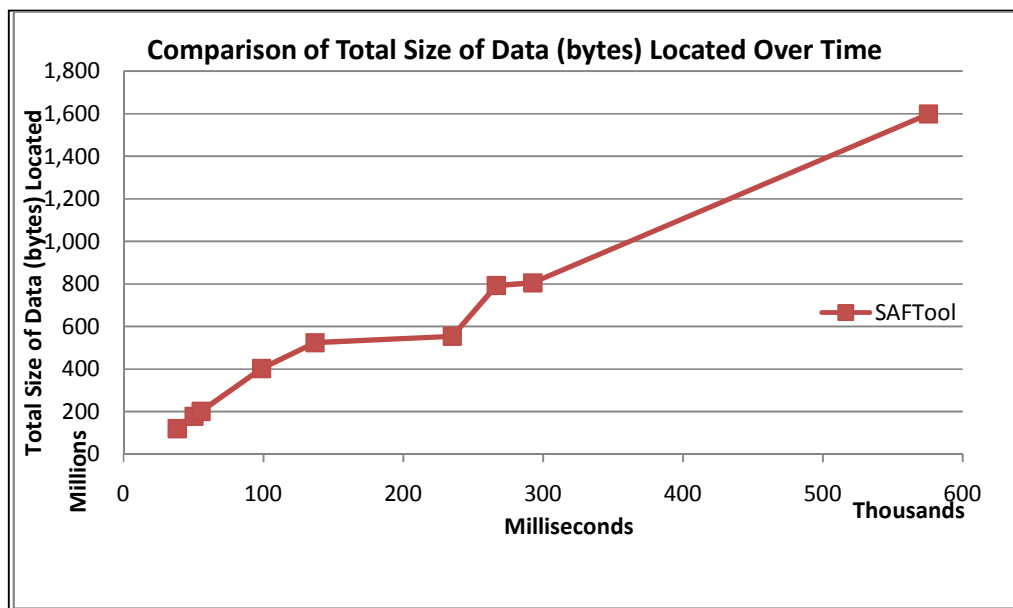


Figure 6.29: Total Size of Data Located Over Time for the SAFTTool

Chapter 2 outlined the key requirements of a forensic investigation including outlining in Chapters 2 and 3 the key features of forensics tools. According to (Volonino et al., 2007), “*computer forensics tools support the investigator by helping to: recreate a specific chain of events or sequence of user activities; search for key words and dates and determine which of the data is relevant; search for copies of previous document drafts; search for potentially privileged information; search for the existence of certain programs, such as file-wiping programs; and authenticate data files and their date and time stamps*”.

According to Shinder and Tittel (2002), swap and page files contain all sorts of data, including e-mail, web pages, word processing documents and any other work that has been performed on the computer during the work session. In line with this, five search terms related to the user Internet activity, specifically e-mail communications were selected to use as a test set for searching. The terms were chosen to check for their existence in the swap files were ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term.

Their existence in the swap files as identified by WinHex and the SAFTool is shown in Table 6.11 and Table 6.12, respectively.

Table 6.11: Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by WinHex

Test Data File Name	Size (KB)	WinHex				
		Existence of 'mail', 'from', 'www', 'html' and 'send' term				
		mail	From	www	Html	send
pagefile01.sys	1,506,576	√	√	√	√	√
pagefile02.sys	540,672	√	√	√	√	√
pagefile03.sys	786,432	√	√	√	√	√
pagefile04.sys	774,144	√	√	√	√	√
pagefile05.sys	117,760	√	√	√	√	√
pagefile06.sys	512,000	√	√	√	√	√
pagefile07.sys	173,648	√	√	√	√	√
pagefile08.sys	196,608	√	√	√	√	√
pagefile09.sys	393,216	√	√	√	√	√

Table 6.12: Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the SAFTTool

Test Data File Name	Size (KB)	SAFTTool				
		Existence of 'mail', 'from', 'www', 'html' and 'send' term				
		mail	From	www	Html	send
pagefile01.sys	1,506,576	√	√	√	√	√
pagefile02.sys	540,672	√	√	√	√	√
pagefile03.sys	786,432	√	√	√	√	√
pagefile04.sys	774,144	√	√	√	√	√
pagefile05.sys	117,760	√	√	√	√	√
pagefile06.sys	512,000	√	√	√	√	√
pagefile07.sys	173,648	√	√	√	√	√
pagefile08.sys	196,608	√	√	√	√	√
pagefile09.sys	393,216	√	√	√	√	√

## 6.5 Subjective Evaluations Experiment of Forensic Analysis System

To validate the results of the tool in interpreting and the data extracted from the Event logs and Swap files, further evaluation experiments were conducted. The objective of the evaluation was: to gain an understanding of the level of awareness surrounding the use of System Generated Artefacts in an investigation; to assess subjects' level of satisfaction with the SAFTool ability to aid their understanding of System Generated Artefacts and their data content; and to assess the degree to which the SAFTool can be modified and extended.

The functionality and usefulness of the SAFTool is further examined by the use of a series of questionnaires. The questionnaires targeted knowledgeable and experienced users in the digital forensics domain. The objectives is to verify the implementation of the SAFTool by (1) viewing different types of events recorded in the Event logs, (2) viewing the contents of Swap files, (3) searching for traces of keywords in Swap files as keyword searching is a primary method for the examination of large blocks of data (Gagnon, 2008), (4) further analysing the event logs and swap files by displaying the file information regarding the contents of the files in a graphical manner, and (5) generating report.

The structure of the experiment is described in Section 6.5.2, which is then followed by a summary of the questionnaire results.

### 6.5.1 Subjects of Experiments

The subjects were recruited from among post-graduate students from the Information Security Research Group, University of Glamorgan. These individuals are in different stages of pursuing doctoral awards and is either actively researching the area of computer forensics or are responsible for teaching on the undergraduate or postgraduate awards offered by the Faculty of Advanced Technology at the University of Glamorgan. The goal of the

system generated artefacts forensic analysis system is to enable it to visualise the known data structure of Event logs and the not known data structure of Swap files in such a way that the investigator can easily see what data is available within these system areas of the Windows, only students who already had previously participated in a disk study were recruited as subjects. Further, because the system is a prototype, that at the moment only handles event logs and swap files, a basic knowledge of event logs and swap files was also required so that subjects could easily understand the tasks set and later be better able to evaluate the data in the display.

The objective of the evaluation was to evaluate the functionalities and usefulness of the prototype architecture with the utilisation of system generated artefacts as forensic evidence.

Test files were extracted from the hard drive images from the forensic laboratory disk study. Only Event logs files and Swap files were extracted as subjects. The experiments used AppEvent04.Evt, SecEvent04.Evt, SysEvent04.Evt and pagefile10.sys from the examined disks (readable disks) that were used in the analysis of information remaining on disks offered for sale on the second hand market at the University of Glamorgan (2008). This is listed in Appendix G.

A total of five subjects participated in the evaluation experiments. All the five subjects had been involved to varying degrees in the residual data disk studies which had examined volumes and types of information that remains on computer hard drives offered for sale on the second hand market. Thus, they had been using system artefacts as forensic object. The subjects' expertise in forensic analysis varied, ranging from specialists in hard drive forensic analysis, mobile forensics to network forensics. Table 6.13 summarises their background knowledge about forensic in general and system artefacts in particular.

As outlined four files subjects were selected in the evaluation experiments and Table 6.14 summarises the properties of the files. Three files is used for the event logs since an event logs comprises three files as outlined



in Chapter 3, and one file is used for the swap file, for analysis and visualising the event logs and swap file.

Table 6.13: Forensic Knowledge and Expertise of Human Subjects

Subject	S1	S2	S3	S4	S5
Years of experience in computer forensics field	5	2	>5	1	5
No. of computer forensics tool used	5	2	>5	1	5

Table 6.14: File Size of File Subjects

Subject	Size (KB)
AppEvent04.Evt	512
SysEvent04.Evt	512
SecEvent04.Evt	64
pagefile10.sys	393,216

### 6.5.2 Structure of Experiments

#### i. Questionnaires:

A questionnaire (see Appendix I) was given to all subjects to be answered after completion of the set task.

Objectives of the questionnaire:

1. To find out whether they were aware of system generated artefacts before this experiment and if so had they examined system generated artefacts to find digital evidence.
2. To find out whether this prototype software helped them easily see what data was available within these system areas of the Windows operating system.

#### ii. The process:

The questionnaires process was as follows:

1. A briefing was given to participants, describing the tasks and objectives of the questionnaire.
2. The test data was explained to participants. They were requested to use the test data for the experiments.

3. The participants were then required to carry out an evaluation of the system generated artefacts (AppEvent04.Evt, SecEvent04.Evt, SysEvent04.Evt and pagefile10.sys) as forensic evidence using the prototype software (SAFTool).
4. The participants were then asked to answer the questionnaire as in Appendix I, which addressed the metrics for performing each type of evaluation.
5. The results from the questionnaires were then analysed and presented in tabulated form and graphical charts.

**iii. The task:**

**A. The Extensible Architecture of the Prototype**

The following tasks were used in the experiments:

**Task 1:**

To verify the extensible architecture, do the following:

1. Click **Analyse** (No displayed artefact name)
2. Click **Visualise** (No displayed artefact name)
3. Click **Report** (No displayed artefact name)

**Task 2:**

To verify the extensible architecture after the Event log plugin and/or Swap file plugin have been installed, do the following:

1. Click **Analyse** (Artefact names are displayed)
2. Click **Visualise** (Artefact names are displayed)
3. Click **Report** (Artefact names are displayed)

Note: This task is used only after the Event log plugin and/or Swap file plugin have been installed.

**B. For Event log files**

The following tasks were used in the experiments:

## Task 1:

To install the event log plugin with SAFTTool do the following:

1. Open SAFTTool and examine the drop-down menu selections.
2. Click **File > Install Plugin**
3. Select **EventLogPlugin.xml > Open** (plugin descriptor for installation)
4. Plugin Installer > **Install Plugin** (Plugin Installer Dialog displayed)
5. Confirmation Dialog > **Yes** (Confirmation to restart base application)

## Task 2:

To open and analyse the event log file with SAFTTool do the following:

1. Click **Analyse > Analyse Event Log**
2. In the File Open dialog, browse to the AppEvent10.Evt file.  
Click **Open**
3. Message popup will appear.

## Task 3:

To open and further analyse the event log file (visualise the event log files) with SAFTTool do the following:

1. Click **Visualise > Visualise Event Log**
2. In the view window, browse to the AppEvent10.Evt file,  
Click **View**
3. To view graph,  
Click **Graph**

## Task 4:

To generate report with SAFTTool do the following:

1. Click **Report > Report Event Log**
2. In the view window, browse to the AppEvent10.Evt file,  
Click **Generate**
3. The Microsoft word documents will popup.

Note: All these tasks can be used for AppEvent10.Evt, SecEvent10.Evt and SysEvent10.Evt.

### C. For Swap files

The following tasks were used in the experiments:

#### Task 1:

To install the pagefile plugin with SAFTTool do the following:

1. Open SAFTTool and examine the drop-down menu selections.
2. Click **File > Install Plugin**
3. Select **PageFilePlugin.xml > Open** (plugin descriptor for installation)
4. Plugin Installer > **Install Plugin** (Plugin Installer Dialog displayed)
5. Confirmation Dialog > **Yes** (Confirmation to restart base application)

#### Task 2:

To open and analyse the pagefile file with SAFTTool do the following:

1. Click **Analyse > Analyse Pagefile**
2. In the File Open dialog, browse to the pagefile10.sys file.  
Click **Open**
3. Message popup will appear.

#### Task 3:

To open and further analyse the pagefile file (visualise the pagefile file) with SAFTTool do the following:

1. Click **Visualise > Visualise Pagefile**
2. In the view window, browse to the pagefile10.sys file,  
Click **Hex View**
3. To view density map,  
Click **Map View**

#### Task 4:

To search word with SAFTTool (keyword search), do the following:

1. Click **Visualise > Visualise Pagefile**
2. In the search window, key in 'mail' word,  
Click **Search**

3. The 'mail' word will be displayed if it exists

Task 5:

To generate report with SAFTTool do the following:

1. Click **Report > Report Pagefile**
2. In the view window, browse to the pagefile10.sys file,  
Click **Generate**
3. The Microsoft word documents will popup.

### 6.5.3 Results

This section reviews and analyses the findings of the evaluation experiments. The objectives of the experiments were to verify the implementation of the SAFTTool through: (1) verifying the extensible architecture; (2) viewing different types of events recorded in the Event logs, (3) viewing the contents of Swap files; (4) searching for traces of keywords in Swap files as keyword searching is the primary examination method (Gagnon, 2008) since swap and page files contain all sorts of data, including e-mail, web pages, word processing documents and any other work that has been performed on the computer during the work session; (5) further analysing the event logs and swap files by displaying the file information relating to the contents of the files in a graphical manner; and (6) generating a report. Developing a plugin was not undertaken in the testing of the architecture's extensibility since in order to build a plugin, the subjects would have been required to analyse each of the programs in the collection and write their own code for the new plugin. Considering that the subjects were volunteers, such an activity would have been time consuming. Five subjects were asked to use the system and verify the functionalities and usefulness of the SAFTTool. Their expert opinion was captured using questionnaires.

The five subjects were given with four files to analyse and visualise. In total, there were 95 tasks conducted with the forensic analysis system during the experiment. A detailed review of these tasks is included in Appendix G. The different tasks were related to assessing different sections of

the architecture: six tasks related to the architecture, forty-five tasks to the Event logs and twenty tasks to the Swap files.

The task with regard to the architecture was to verify that the architecture is extensible; which new plugins of various Windows artefacts can be added and with various visualisation techniques, which can be used for the analysis and visualisation of the artefacts. The tasks with regard to the Event logs is to view different types of events recorded in the Event logs; visualise the file information regarding the contents of the files in graphical manner, which can aid the investigator to interpret massive amounts of data and representations of events in chronological order helps understanding much easier; and generating report, which is used when the investigator requires the information from the analysis to be included in a report. The tasks with regard to the Swap files is to view the contents of the Swap files; searching for traces of keywords since swap and page files contain all sorts of data, including e-mail, web pages, word processing documents and any other work that has been performed on the computer during the work session; visualise the file information regarding the contents of the files in block view format (nonhierarchical visualisation technique) which can aid the investigator to interpret massive amounts of data and this helps understanding much easier; and generating report, which is used when the investigator requires the information from the analysis to be included in a report. Upon conducting a task, the user would see the outcome of the task in a different format: the Event logs displayed data in one line per entry and graphical form, i.e. in table and timeline; and Swap files displayed information in hexadecimal data display and ASCII text display and square block view format which display the file information regarding of file content.

At the same time, the subjects' opinions regarding the functionalities and usefulness of the SAFTool were gathered through the comment columns of the questionnaires. Their contents are summarised below:

(1) Features to help users:

Menu, colours and presentation are elements to help users in using the tool and understanding the objectives of the usage of the tool. Use of

colours to identify different areas and to easy understand visual aids will also help. The SAFTool should have a 'Help' menu for users to understand how the tool works.

(2) Information presentation:

Visualisation is the key element to aid users' exploration of the file contents. The SAFTool presents a visual presentation of artefacts generated by the system both graphically and textually. Presentation of information was in a structured manner but with a basic level of detail. Start with Analyse, Visualise then Report. With the search function, the possibility of any queries can be seen through the system generated artefacts and therefore supply a better understanding. In order to better understand the information in the files the SAFTool should have more information on visual representation. For example, incorporate better description for the graph's legend by explaining more of the units used in the block diagram of the Swap files.

(3) Consideration of interface design:

The most important element for the interface design is the menu structure. The SAFTool's linear menu is easy to operate and demonstrate the flow design of left to right in the menus. The SAFTool, at the same time, uses branched menus in one visual representation of the file's content to branch out to another visual representation of the file's content.

(4) Enhancement for expansion:

The SAFTool has a built-in extensible architecture. It seems the tool could be expanded with additional add ons. It is a straightforward process, just a couple of clicks and a plugin is installed, but this needs to be further investigation.

In order to study the functionalities and usefulness of system generated artefacts as forensic evidence, the functionalities and usefulness of the prototype software with additional information annotation as a working tool; and the SAFTool's built-in extensible architecture, allowing it to be expanded

with additional add on, three objectives were introduced and analysed as follows:

- O1: To gain understanding of the level of awareness surrounding the use of System Generated Artefacts in an investigation;
- O2: To assess the degree to which the SAFTool supports the understanding of System Generated Artefacts; and
- O3: To assess the degree to which the SAFTool can be modified and extended.

The three objectives were assessed using the questionnaire distributed to the five subjects. Their satisfaction levels will the objectives established in questions A1, A2, A3, B4, B5 and C6 below were analysed:

- A1: The data analysis and the presentation of data in the application aid the understanding of the selected system generated artefacts. The use of the tool highlights the importance of system generated artefacts.
- A2: The chosen analysis and display methods aid the understanding of the system generated artefacts data.
- A3: Additional information annotation provided in the application aids the understanding of the information contained in the system generated artefacts.
- B4: The data analysis and data visualisation easily and clearly indicate the contents of the Windows system generated artefacts.
- B5: The data analysis and data visualisation used in the prototype aid in understanding the information contained in the Windows system generated artefacts.
- C6: In your expert opinion, and given the information provided, how easy is it to modify and extend the tool?

In Figure 6.30, participants' satisfaction levels are tabulated and presented in the form of a bar chart with categories relating to the functionalities and usefulness of the system generated artefacts as forensic evidence; the functionalities and usefulness of the prototype software with additional information annotation as a working tool; and the SAFTool's built-in



extensible architecture allowing it to be expanded with additional add ons. The questionnaire for the experiments undertaken is provided in Appendix I.

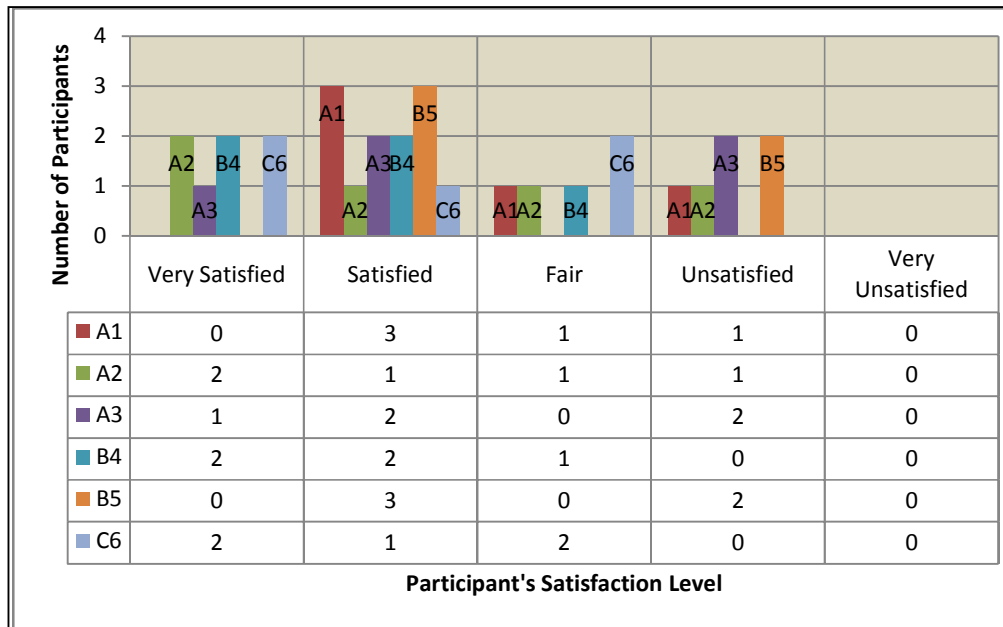


Figure 6.30: Participant's Satisfaction Level with the System Generated Artefacts Forensic Analysis Tool (SAFTool)

Figure 6.30 above depicts the results of the participant's satisfaction level on the satisfaction of analysing, visualising and reporting system generated artefacts concentrating on Event logs and Swap files which is built in the extensible architecture of SAFTool. Generally, majority of the participants are in the fair and above fair category of satisfaction with the functionalities and usefulness on the use of System Generated Artefacts in an investigation and the support of System Generated Artefacts Forensic Analysis Application (SAFTool) in understanding of System Generated Artefacts.

Looking at the subjects' satisfaction level with the objectives established in questions A1, A2, A3, B4, B5 and C6, depicted by the bar chart, more than 50% of the subjects expressed a fair and above level of satisfaction with them: 4(80%) with A1; 4(100%) with A2; 3(60%) with A3, 5(100%) with B4, 3(60%) with B5; and 5(100%) with C6. Subjects thus indicated they were

satisfied with their ability to analyse, visualise and report on system generated artefacts using an extensible architecture during an investigation.

Looking at subjects who were unsatisfied with the objectives established in questions A1, A2, A3, B4, B5 and C6, depicted by the bar chart, less than 50% were unsatisfied: 1(20%) with A1; 1(20%) with A2; 2(40%) with A3, 0(0%) with B4, 2(40%) with B5; and 0(0%) with C6. Not one subject was very unsatisfied.

The result thus show and confirm that the SAFTool aided understanding system generated artefacts; the additional information annotation provided in the tool aided understanding of the information contained in the system generated artefacts; the data analysis and visualisation functionalities easily and clearly indicated the contents of the artefact; and the SAFTool's built-in extensible architecture would allow it to be expanded with additional add ons. Hence all the objectives: O1 (to gain an understanding and increase the level of awareness surrounding the use of system generated artefacts in an investigation); O2 (to assess subjects' level of satisfaction with the SAFTool's ability to aid their understanding of system generated artefacts); and O3 (how easy they thought it would be to modify and extend the tool) were accomplished.

## 6.6 Additional Experiments

The testing of the tool for consistency is examined here via a several number of test files would be used and number of times that each one was tested, to demonstrate that a tool meets the consistency test. For this test, all the experiments were performed three times run on a number of artefacts files of varying sizes and were extracted from different machine (dataset from Appendix H) and using the tools mentioned earlier in Section 6.2. A new database was created each time of the run.

The result of applying the various tools to the test files listed in Appendix H is summarised for discussion in Table 6.15. This table indicates the number of records that are processed from individual event logs (one line per log file) from the three runs of each event logs. Figure 6.31 illustrates the total event records from three runs of each event logs in Appendix H located for the evtstats.pl script and the prototype SAFTTool in the consistency test.

Table 6.16 shows the average time required to process each event logs using lsevt.pl and SAFTTool (from three runs of each event logs in Appendix H). The graph in Figure 6.32 is generated based on the parses through the contents of the Event log file and display event records. The total processing time would be in milliseconds and the practical impact is that extra in average 1521 milliseconds are taken. Both illustrate almost the same shapes of graph, i.e. processing time increases with the file sizes. As noted in Section 6.4.7, the experiment with the Event Viewer could not be conducted.

Table 6.15: Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTool in Consistency Test

Test Data File Name	Size (KB)	No. of Records		
		Event Viewer	evtstats.pl	SAFTool
AppEvent05.Evt	64	X	112	112
SecEvent05.Evt	64	X	58	58
SysEvent05.Evt	128	X	347	347
AppEvent06.Evt	64	X	76	76
SecEvent06.Evt	64	X	0	0
SysEvent06.Evt	64	X	262	262
AppEvent07.Evt	64	X	123	123
SecEvent07.Evt	64	X	260	260
SysEvent07.Evt	256	X	370	370
AppEvent08.Evt	192	X	139	139
SecEvent08.Evt	64	X	0	0
SysEvent08.Evt	512	X	1547	1547
AppEvent09.Evt	192	X	106	106
SecEvent09.Evt	512	X	2317	2317
SysEvent09.Evt	128	X	395	395
AppEvent10.Evt	64	X	63	63
SecEvent10.Evt	64	X	0	0
SysEvent10.Evt	64	X	295	295
AppEvent11.Evt	64	X	187	187
SecEvent11.Evt	64	X	0	0
SysEvent11.Evt	128	X	359	359
AppEvent12.Evt	64	X	110	110
SecEvent12.Evt	64	X	206	206
SysEvent12.Evt	64	X	236	236
AppEvent13.Evt	64	X	89	89
SecEvent13.Evt	64	X	0	0
SysEvent13.Evt	64	X	247	247
AppEvent14.Evt	64	X	76	76
SecEvent14.Evt	64	X	124	124
SysEvent14.Evt	128	X	312	312

X – unable to complete the operation. The event log file is reported as corrupted

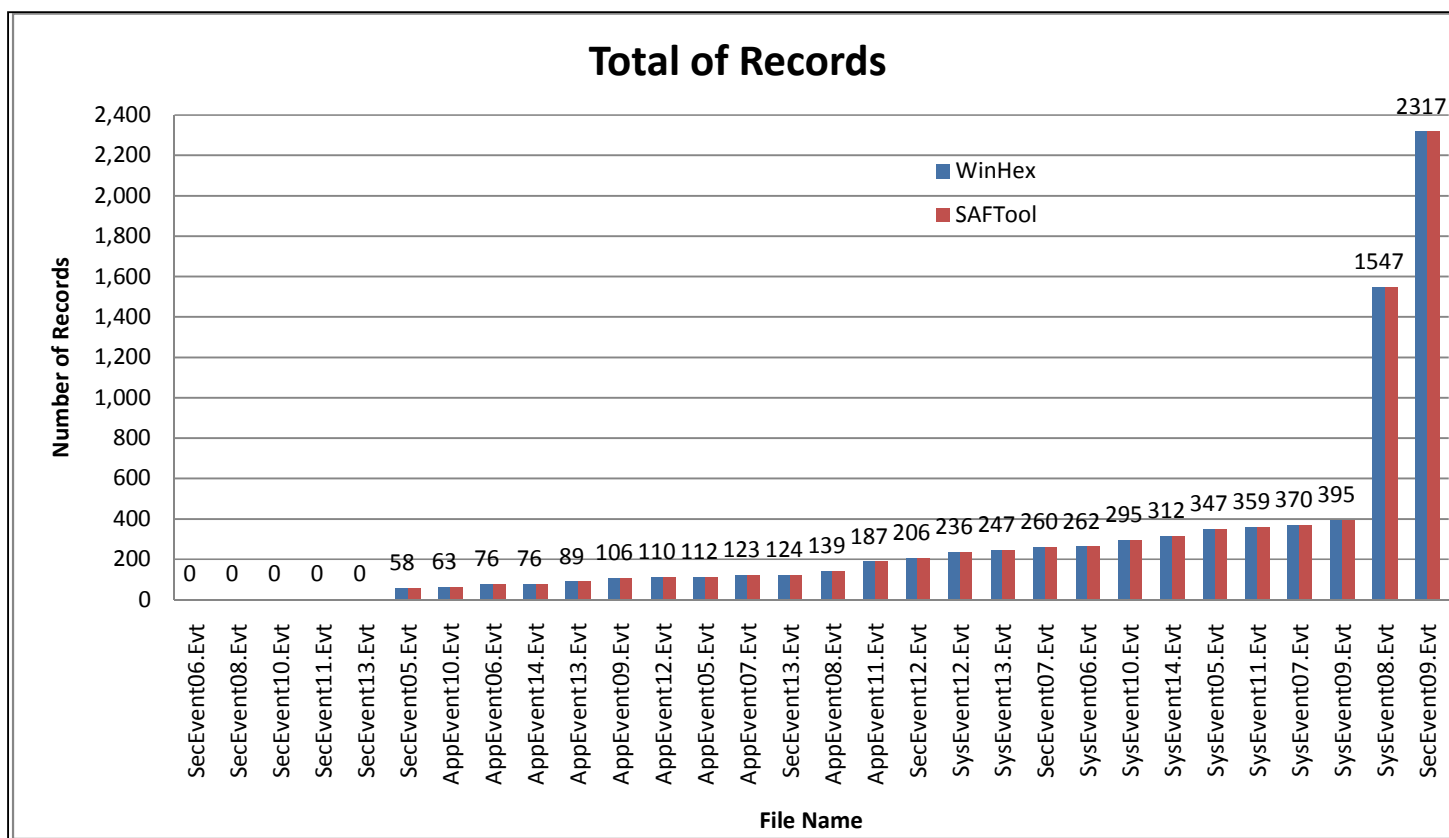


Figure 6.31: Total Event Records Located for the evtstats.pl Script and SAFTool in Consistency Test

Table 6.16: Processing Time for Event Viewer, lsevt.pl and SAFTool in Consistency Test

Test Data File Name	Size (KB)	Event Viewer Processing Time (ms)	lsevt.pl		SAFTTool	
			No. of Records	Processing Time (ms)	No. of Records	Processing Time (ms)
AppEvent05.Evt	64	X	112	87	112	661.33
SecEvent05.Evt	64	X	58	57	58	343.66
SysEvent05.Evt	128	X	347	198.66	347	1390.66
AppEvent06.Evt	64	X	76	70	76	437.33
SecEvent06.Evt	64	X	0	0	0	0
SysEvent06.Evt	64	X	262	149.33	262	1239.33
AppEvent07.Evt	64	X	123	87.66	123	677
SecEvent07.Evt	64	X	260	148	260	1089
SysEvent07.Evt	256	X	370	396.66	370	1729
AppEvent08.Evt	192	X	139	104.66	139	838.33
SecEvent08.Evt	64	X	0	0	0	0
SysEvent08.Evt	512	X	1547	804.66	1547	6025.66
AppEvent09.Evt	192	X	106	79.33	106	474
SecEvent09.Evt	512	X	2317	851	2317	9755.33
SysEvent09.Evt	128	X	395	397.33	395	2805.66
AppEvent10.Evt	64	X	63	60	63	349
SecEvent10.Evt	64	X	0	0	0	0
SysEvent10.Evt	64	X	295	150	295	1251
AppEvent11.Evt	64	X	187	115.66	187	874.66
SecEvent11.Evt	64	X	0	0	0	0
SysEvent11.Evt	128	X	359	263.33	359	1583.66
AppEvent12.Evt	64	X	110	85	110	531.66
SecEvent12.Evt	64	X	206	118.33	206	901
SysEvent12.Evt	64	X	236	136.33	236	911.66
AppEvent13.Evt	64	X	89	75.66	89	468.66
SecEvent13.Evt	64	X	0	0	0	0
SysEvent13.Evt	64	X	247	147.66	247	1067.66
AppEvent14.Evt	64	X	76	67	76	401.33
SecEvent14.Evt	64	X	124	97	124	828
SysEvent14.Evt	128	X	312	162	312	1385.33

X – unable to complete the operation. The event log file is reported as corrupted.

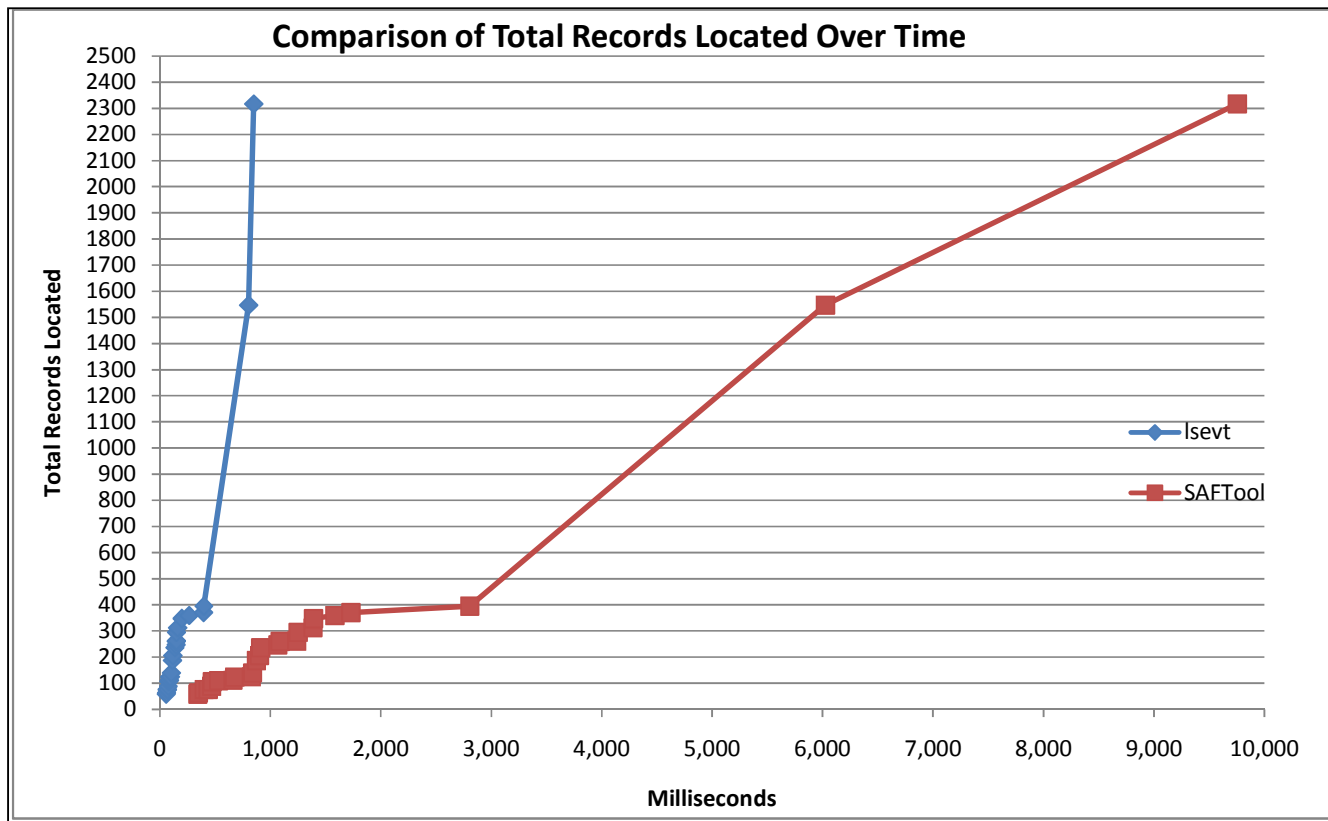


Figure 6.32: Graph Showing the Number of Records Located With Processing Time in Consistency Test

Data in Table 6.17 shows total size of data parsed for WinHex and the SAFTool (from three runs of each swap file in Appendix H). As can be seen, the same results were recorded for each tool in the consistency test. A graph (see Figure 6.33) was subsequently drawn comparing the total size of data parsed by each tool. It confirmed the result in Table 6.17.

Table 6.17: Total Size of Data for WinHex and the SAFTool in Consistency Test

Test Data File Name	Size (KB)	Total Size of Data (bytes)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	1,598,029,824	1,598,029,824
pagefile12.sys	589,824	603,979,776	603,979,776
pagefile13.sys	2,095,104	2,145,386,496	2,145,386,496
pagefile14.sys	786,432	805,306,368	805,306,368
pagefile15.sys	196,608	201,326,592	201,326,592
pagefile16.sys	540,672	553,648,128	553,648,128
pagefile17.sys	117,760	120,586,240	120,586,240
pagefile18.sys	512,000	524,288,000	524,288,000
pagefile19.sys	774,144	792,723,456	792,723,456
pagefile20.sys	393,216	402,653,184	402,653,184



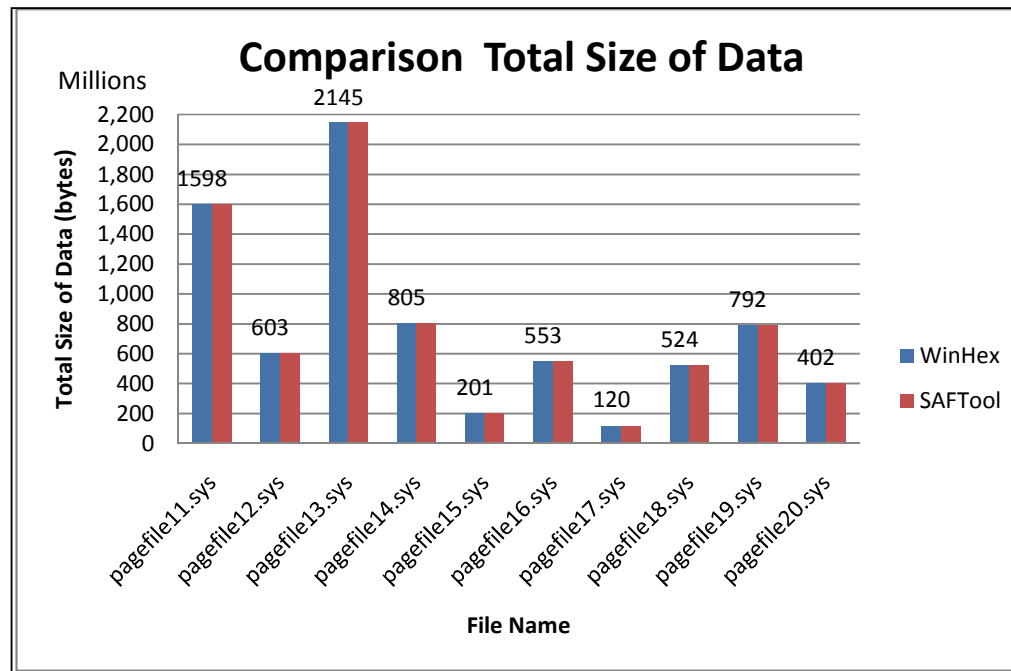


Figure 6.33: Comparison of Total Size of Data Shown by the WinHex and SAFTTool for Consistency Test

Table 6.18 shows the processing times for WinHex and the SAFTTool (from three runs of each swap file in Appendix H) which was subsequently displayed in graph form, Figure 6.34 and Figure 6.35 respectively.

In the context of this chapter, processing time recorded in milliseconds (ms) in term of the time taken by the forensic analysis system to generate an analysis display. It took a longer generate hexadecimal data display or an ASCII text display for the contents of each swap file when the SAFTTool was employed in the consistency test.

Table 6.18: Processing Time for WinHex and SAFTTool in Consistency Test

Test Data File Name	Size (KB)	Processing Time (ms)	
		WinHex	SAFTTool
pagefile11.sys	1,506,576	23.33	461,594
pagefile12.sys	589,824	21.00	299,812
pagefile13.sys	2,095,104	25.00	572,125
pagefile14.sys	786,432	22.66	359,896
pagefile15.sys	196,608	19.33	234,735
pagefile16.sys	540,672	20.66	295,911
pagefile17.sys	117,760	18.33	228,339
pagefile18.sys	512,000	20.33	290,052
pagefile19.sys	774,144	21.33	338,813
pagefile20.sys	393,216	19.66	283,339

The ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term existence in the swap files (from three runs of each swap file in Appendix H) as identified by WinHex and the SAFTTool is shown in Table 6.19 and Table 6.20 respectively.

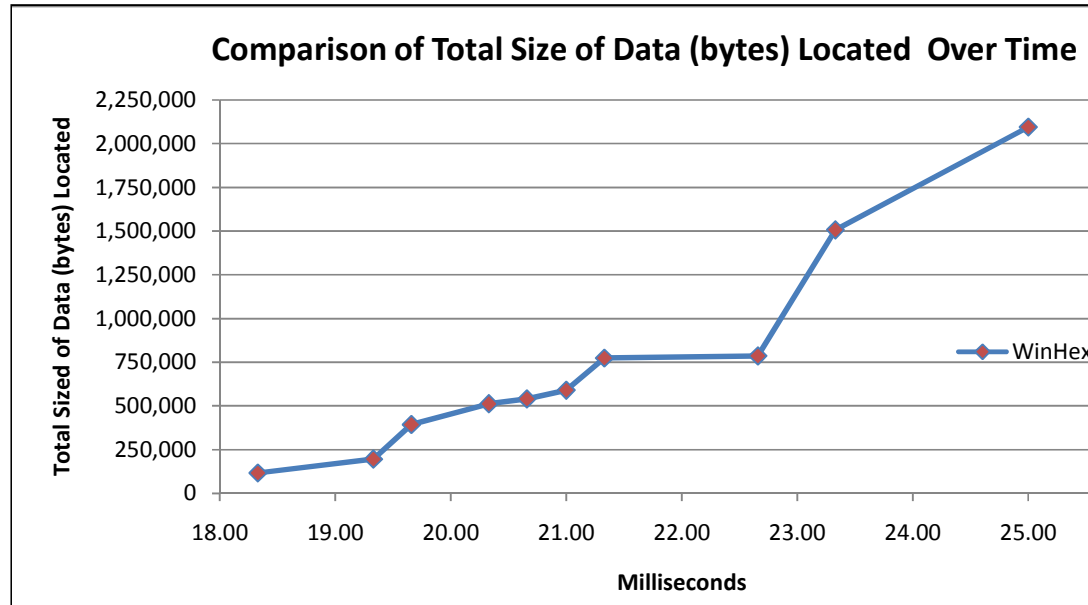


Figure 6.34: Total Size of Data Located Over Time for WinHex in Consistency Test

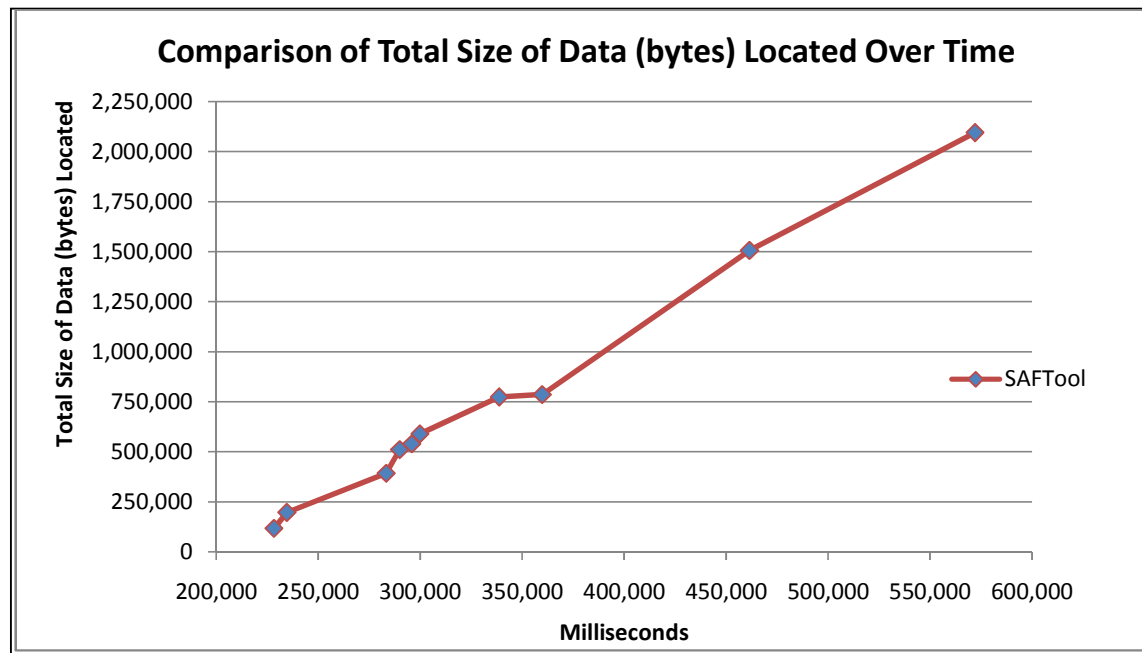


Figure 6.35: Total Size of Data Located Over Time for the SAFTTool in Consistency Test

Table 6.19: Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by WinHex in Consistency Test

Test Data File Name	Size (KB)	WinHex				
		Existence of 'mail', 'from', 'www', 'html' and 'send' term				
		mail	From	www	Html	send
pagefile11.sys	196,608	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	1,560,576	√	√	√	√	√
pagefile14.sys	589,824	√	√	√	√	√
pagefile15.sys	393,216	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	786,432	√	√	√	√	√

Table 6.20: Existence of 'mail', 'from', 'www', 'html' and 'send' Terms in Swap Files Identified by the SAFTTool in Consistency Test

Test Data File Name	Size (KB)	SAFTTool				
		Existence of 'mail', 'from', 'www', 'html' and 'send' term				
		mail	From	www	Html	send
pagefile11.sys	196,608	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	1,560,576	√	√	√	√	√
pagefile14.sys	589,824	√	√	√	√	√
pagefile15.sys	393,216	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	786,432	√	√	√	√	√

A total of 15 subjects participated in the evaluation experiments to evaluate the tool. This number is three times the number of subjects involved in evaluating the tool as undertaken in Section 6.5.3, therefore should show a realistic population. All the 15 subjects involve directly in digital forensic examination and analysis including in the area of audio and video forensics. Thus, they had been using system artefacts as forensic object. The subjects' expertise in forensic analysis is hard drive forensic analysis, phone forensics, audio forensics and video forensics. Table 6.13 summarises their background knowledge about forensic in general and system artefacts in particular.

Test files were extracted from the hard drive from the multimedia laboratory used to train computer technician. Only Event logs files and Swap files were extracted as subjects. The experiments used AppEvent05.Evt, SecEvent05.Evt, SysEvent05.Evt and pagefile20.sys in the analysis. This is listed in Appendix H. As outlined four files subjects were selected in the evaluation experiments and Table 6.14 summarises the properties of the files.

Table 6.21: Forensic Knowledge and Expertise of Human Subjects of Fifteen Participants

(a) Subject 1 to 5

Subject	S1	S2	S3	S4	S5
Years of experience in computer forensics field	5	2	2	1	5
No. of computer forensics tool used	9	2	4	2	5

(b) Subject 6 to 10

Subject	S6	S7	S8	S9	S10
Years of experience in computer forensics field	5	2	6	1	4
No. of computer forensics tool used	6	2	7	1	5

(c) Subject 11 to 15

Subject	S11	S12	S13	S14	S15
Years of experience in computer forensics field	5	2	>5	1	5
No. of computer forensics tool used	5	2	>5	1	5

Table 6.22: File Size of File Subjects for Additional Experiments

Subject	Size (KB)
AppEvent05.Evt	64
SysEvent05.Evt	128
SecEvent05.Evt	64
Pagefile20.sys	393,216

The 15 subjects were given with four files to analyse and visualise. In total, there were 285 tasks conducted with the forensic analysis system during the experiment. A detailed review of these tasks is included in Appendix H (Table H4). The different tasks were related to assessing different sections of the architecture: 90 tasks related to the architecture, 135 tasks to the Event logs and 60 tasks to the Swap files. The outline of the experiment has already been discussed in Section 6.5.2.

This section contains the results of the questionnaires as initially discussed in Section 6.5.3. In Figure 6.36, participants' satisfaction levels are tabulated and presented in the form of a bar chart with categories relating to the functionalities and usefulness of the system generated artefacts as forensic evidence; the functionalities and usefulness of the prototype software with additional information annotation as a working tool; and the SAFTool's built-in extensible architecture allowing it to be expanded with additional add ons. The questionnaire for the experiments undertaken is provided in Appendix I.

Figure 6.36 depicts the results of the participant's satisfaction level on the satisfaction of analysing, visualising and reporting system generated artefacts concentrating on Event logs and Swap files which is built in the extensible architecture of SAFTool. Generally, all of the participants are in the fair and above fair category of satisfaction with the functionalities and usefulness on the use of System Generated Artefacts in an investigation and the support of System Generated Artefacts Forensic Analysis Application (SAFTool) in understanding of System Generated Artefacts.

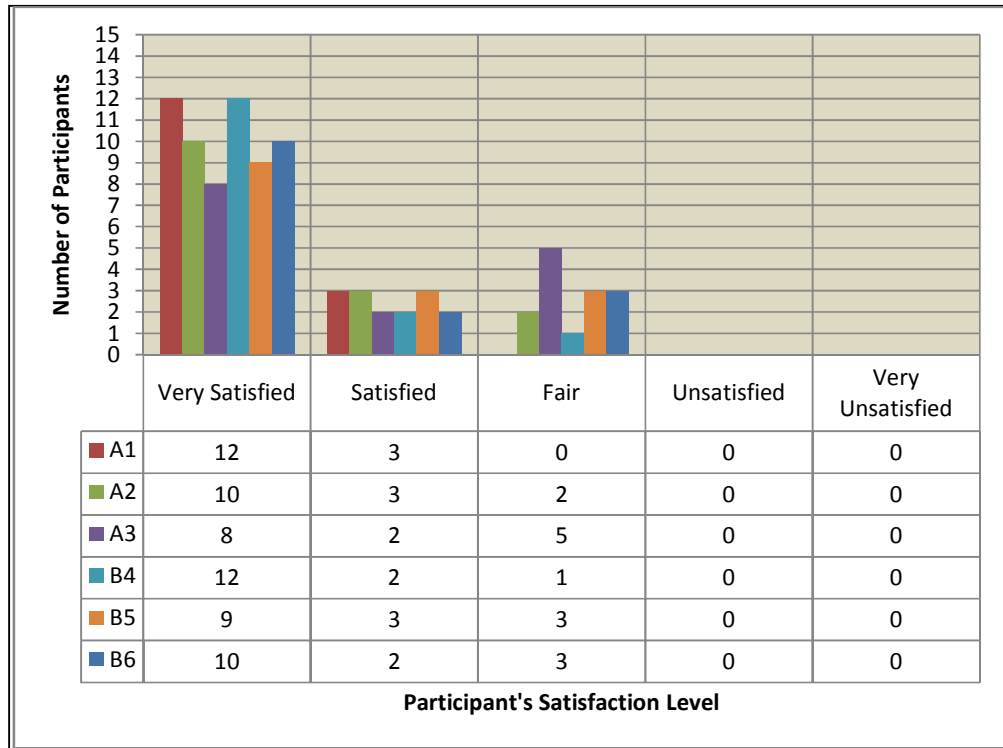


Figure 6.36: Participant's Satisfaction Level with Fifteen Participants in Using the System Generated Artefacts Forensic Analysis Tool (SAFTTool)

Looking at the subjects' satisfaction level with the objectives established in questions A1, A2, A3, B4, B5 and C6 depicted by the bar chart, more than 50% of the participants are very satisfied. Out of the total of 15 participants, 12 (80%) of A1, followed by 10 (67%) of A2, 8 (53%) of A3, 12 (80%) of B4, 9 (60%) of B5, and lastly 10 (67%) of B6 show that by using visualised tool, system generated artefacts can be easily read to enhance the understanding. Subjects thus indicated they were satisfied with their ability to analyse, visualise and report on system generated artefacts using an extensible architecture during an investigation. Whereas, the percentage of below 50% on the participant's satisfaction level of satisfied, fair, 0% for unsatisfied and very unsatisfied are described as follows:



- i. Satisfied (3 (20%) of A1; 3 (20%) of A2; 2(13%) of A3; 2(13%) of B4; 3(20%) of B5; and 2(13%) of B6)
- ii. Fair (0 (0%) of A1; 2 (13%) of A2; 5(33%) of A3; 1(7%) of B4; 3(20%) of B5; and 3(20%) of B6)

The result thus show and confirm that the SAFTool aided understanding system generated artefacts; the additional information annotation provided in the tool aided understanding of the information contained in the system generated artefacts; the data analysis and visualisation functionalities easily and clearly indicated the contents of the artefact; and the SAFTool's built-in extensible architecture would allow it to be expanded with additional add ons. Hence all the objectives mentioned in Section 6.5.3 were accomplished.

To conclude, the SAFTool (the visualised tool) with built-in extensible architecture and with additional information annotation (menu, colours and presentation) as a working tool have enhanced the understanding about system generated artefacts.

## 6.7 Conclusion

In this chapter we have presented the results of a number of comparisons and evaluations of the proposed system generated artefacts forensic analysis system. Based on the experiments undertaken, the proposed system to locate and extract data from the data structures is more robust and accurate than approach using the Windows API to analyse system generated artefacts. Further, the system has been shown able to locate and extract data from a known and a not known data structures and present users with the data using different visualisation techniques. We obtained approximately 100% of records examining both event logs (parsing the contents of the Event log file and performing an actual count of the number of records found and not parses the header of the Event log file and determines the number of records that existed

to both evtstats.pl and SAFTool) and swap files (equal total size of data parsed given a pagefile to both WinHex and SAFTool). According to Teerlink and Erbacher (2006), the forensic analysis process can be made better if visualisation is employed to display mountains of data in analysing suspicious files.

In addition we also learned that there is a difference in processing time in locating and extracting known and not known data structures. Based on the experiments undertaken, it was noted that swap files (with a not known data structure) took a longer processing time when compared to event logs files (with a known data structure). This was probably due to the difference in their data structure, i.e. one known and one not known.

The results produced by the questionnaire evaluation confirmed the functionalities and usefulness of system generated artefacts as forensic evidence; the functionalities and usefulness of the prototype software with additional information annotation as a working tool; SAFTool's built-in extensible architecture allowing it to be expanded with additional add ons; and the data analysis and data visualisation used in the prototype increasing understanding. Generally, all of the participants (five and fifteen) are in fair and above fair category of satisfaction with the functionalities and usefulness on the use of System Generated Artefacts in an investigation and the support of System Generated Artefacts Forensic Analysis Application (SAFTool) in understanding of System Generated Artefacts. 100% of records obtained examining both event logs and swap files with the three times that each file was tested demonstrate the consistency of the tool.

## CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

In the introduction to this thesis, a hypothesis was proposed and a series of objectives were outlined which focused on testing the hypothesis. These were concerned with the tools, processes and procedures required to support the forensic analysis of Windows system generated artefacts. This chapter reflects on these outlined objectives, describe how the various chapters contribute to addressing the questions, raised and also draws a series of conclusions.

### 7.1 Summary

The purpose of the thesis was to prove that the prototype tool could:

- Benefit investigators by providing a mechanism through which the analysis and visualisation of system artefacts can effectively reveal the contents of a selection of system artefacts to further assist investigators in examining evidence.
- Allow investigators to see the evidentiary value of each of the selected system generated artefacts as they incorporate new modules of analysis, visualisation and report for those artefacts.

- Assist in the process of examining system artefacts, by incorporating search for keywords function and visualisation techniques to interpret massive amounts of data and this helps understanding much easier.

In the literature review contained in (Chapters 2 and 3), a need was identified in the lack of special software to analyse system artefact data in order to find information related to a specific case, which data can be found in many different places. Several of these artefacts are created on Windows systems during normal operations without reference to the user and without the user's knowledge. System generated artefacts represent valuable sources of evidence and are increasingly the focus of investigation and legal discovery as they are generated by the system and are not readily visible to the common user, which also makes it more plausible that they have not been altered (Volonino et al., 2007).

Further, with ever increasing hard drive capacity, and also for critical business requirements it is more practical to be able to gain access to the file(s) required since, as pointed out by Carvey (2004), *"I found that in many ways, all these forensic analysis applications are vastly different, different in the capabilities they provide, and especially different in how you would go about getting them to perform a certain function and then display the results. A forensic analysis application needs a core set of functionalities and capabilities. A forensic analysis application should be a data presentation application."* This statement would appear to support the development of new forensic analysis application.

The aim and objectives of the research were influenced by a literature review and by observing and using the current state of the art tools available for the analysis of Windows system generated artefacts. The research was structured around three central questions 1 to 3 summarised below which in Section 7.1.1 through Section 7.1.3, and which have been addressed and stated by the summarisation of related chapters. This further led to the architecture design and implementation of the prototype tool (SAFTool)

(Chapters 4 and 5). Next, coupled with the research presented in this thesis, a new solution to a problem has been evaluated (Chapter 6).

In the introduction to this thesis (Chapter 1), a number of questions is posed concerning with the tools, processes and procedures required to support the forensic analysis of Windows system generated artefacts. In this chapter, it will reflect on these questions, describe how the various chapters contribute to answering each question and draw some conclusions.

### 7.1.1 Information Extraction

*Question 1: How can we extract information from an artefact?*

One of the first steps in a forensic analysis is to extract the information contained in an artefact: in this case, the automated extraction of information from an artefact file. In Chapter 1, we argue that this step is hindered in some cases by the internal structure of the file not being generally available. The internal structures being effectively unknown makes it difficult to parse the file for the file evidentiary values. Hence, this project suggested implementing the ideas of abstraction to data structuring and providing appropriate parsers to extract different types of data structure that are explicitly and implicitly contained in a file. Chapter 3 addressed the data structures contained in a number of system generated artefacts. Event logs and Swap files data structures were selected as specific examples for this thesis, as they are examples of the complex internal structures of system artefacts found in the Windows Operating System, Event logs for their known data structure and Swap files for their not known data structure. Chapter 4 introduced the proposed architecture aimed at processing Windows system artefacts. The architecture requirements and objectives were defined based on a clear understanding of the priority of the requirements identified in Chapters 2 and 3 from the issues associated with current state of the art tools. In order to make the architecture extensible, Chapter 4 elaborated on how the design

process of the proposed architecture incorporated and implemented object oriented features, such as abstraction, modular software, event-driven programming, reusable software and model transition. In addition, the architecture has a layered architecture, that is, has separate layers for the application logic, presentation, domain and database, and each layer has different classes. Therefore, this architecture provides an extensibility functionality whereby the application can be extended easily without modifying its original code base. Following the design process was implementation of the prototype system, which dealt with how the design decisions were implemented with respect to the requirements in Chapter 4. Chapter 5 focused on the implementation of the proposed architecture and the development of the prototype and discussed the various techniques used to implement the architecture. In Chapter 6, the prototype implementation and, in turn, the architecture was evaluated using a series of experiments based on test data consisting of previously unseen event log files and swap files.

The architecture deals with the complex structure in the native format. It facilitates the extraction of data from complex structures. The architecture use data structures to access and extract all the relevant data from the complex structures. This can be done for all structures, as it does not use the APIs internally for accessing, adding and retrieving data.

### 7.1.2 Organising Data

**Question 2:** *How can we organise and integrate the various data structures to improve the correlation of the data obtained?*

In Chapter 4, we introduced the requirements for the storage of persistent data to operate independently of any changes made to the database, thus reinforcing the architecture's extensibility. In Chapter 5, we have demonstrated the use of a Database Management System (DBMS) to integrate information on the known and not known data structure of an artefact as

Event logs represent known data structure, whereas swap files represent not known data structure. The database can be used to suit the storage of data for known and not known data structures of Windows system generated artefacts and can be used further for data from different systems. Both file and database provide stores for information but the file for keeping data and the other database design is not used as the file is only appropriate for simple applications and for storing data that does not need to be shared and updated by many users; other database designs tested did not include the use of recursion and resulted in more fields being added to the database records. Therefore, a relational database represents a storage better compared to using files as a means of persistent storage. The DBMS is a comprehensive data management system which organises and manages the tasks associated with storing and providing effective access to large volumes of data. It can be expanded in terms of accepting all sizes of data without any knowledge of the structure and extracting the data for visualisation and report printout purposes.

The architecture enables the automatic processing of complex data structures into a common data store. The data is pulled from the complex structure and stored in the database for use by the visualiser. The architecture also incorporates a mechanism to ensure that no data in the original complex structure will change, to ensure that forensic integrity is maintained during the investigation.

### 7.1.3 Supporting Forensic Analysis

**Question 3:** *How can we use the information obtained in the first two questions to support and improve forensic analysis?*

An issue which arises after parsing and storing the information contained in an artefact is presenting it usefulness, in a way that can be easily understood and interacted with. Visual representations translate data into a visible form

that highlights important features, including commonalities and anomalies (Guillermo et al., 2007). The work undertaken was demonstrated in Chapters 4 and 5 included the operation of mapping data into a visual form such that the data representation is invoked during the use case `Visualise Artefact Data`. The use case description is as follows: *the artefact data may be examined to identify valuable meaningful data and allow for further analysis. The mapping data to a visual form is presented in a more human friendly format by graphs, charts or illustrations*. Results from the design of the use case `Visualise Artefact Data` in Chapter 5 showed that by information visualisation, visual forensic analysis can be improved. Teerlink and Erbacher (2006) commented, “Using this concept of visual perception, we have developed a GUI and associated visualisations that display file information in a graphical manner; and with these visualisation techniques reduce the time examiners need to analyse data and greatly increase the probability of locating criminal evidence”. In Chapter 6, we learnt that the Carvey’s (2007) programs (`evtstats.pl` and `lsevt.pl`) shows the numbers of record in numeric form and the `SAFTool` program shows the number of records in visualisation form in the shape of graph (bar graphs).

The architecture presents the data extracted from the structure visually in many formats using analytical visualisation techniques or informative visualisation techniques. The architecture uses the data stored in the complex structure and uses the visualiser to visualise the output of the data. The visualiser can provide visualisations of data in informative ways, illustrating the data in graphs and charts. Analytical visualisation can provide the user to further analyse the data. The architecture also provides the investigator with a reporting feature. A reporting feature can be used to export out the data to a report.



## 7.2 Conclusions

The following achievements were the results of proving the hypothesis and fulfilling the research aim and objectives outlined in Chapter 1 which allowed verification through a thorough evaluation process:

- A design overview and detailed design with design documentation which detailed in the justifications the decisions taken when discussing the features and development practices required for the implementation of the SAFTool to be a success.
- The implementation of the design, which is the development of an extensible architecture that enables a variety of system artefacts to be analysed by allowing other objects and tools to be added easily. The architecture also caters for different data structures (known and not known internal data structures of system generated files by the Windows Operating System). This includes the provision of visualisation that displays file contents information in a graphical manner.
- The implementation of a proof of concept prototype tool capable of visualising the Event logs and Swap files.
- A detailed evaluation plan was produced. A number of comparable applications from academia and industry were selected as representatives of current state-of-the-art tools and compared to the proposed prototype tool.
- The evaluations carried out produced detailed evaluation results which have been examined and discussed in depth. Conclusions regarding the architecture's ability to satisfy the requirements identified at the start of the research were drawn from the experiments and helped to confirm the contributions made to science.

The architecture developed has satisfied the hypothesis of the thesis. The following are the contributions in more detail:

- An architecture as a single standard means for examining system artefacts contained in hard disk images.
- An architecture as a single standard means for analysing and visualising evidence from various artefacts within the Windows Operating System. These artefacts' data are parsed separately and later combined into a single representation. Such an approach can be extended to include other data from various areas within the Windows operating system as identified and/or required by the user. The data in the complex data structures within the Windows operating system can be easily analysed using the developed architecture.
- An architecture that parses the information in those files in a manner that does not rely on the Windows API.
- Architecture capable of visualising data contained in an artefact in such a way that the investigator can easily see what data what is available within these areas of the Windows operating system.
- An architecture that incorporates reporting functionality which some tools do not offer a reporting facility.
- The provision of an open source architecture that is as extensible and flexible as possible for future improvements, research and the addition of features as well as integration of the implemented system into a wider forensic tool.
- A prototype tool for visualising the Event logs and Swap files' data as a means to highlight its features in order of their importance, reveal patterns, and simultaneously show features that exist across multiple dimensions. Thus, visualisation is the important method to understand and communicate information.

### 7.3 Issues and Future Work

While this research proved that it is feasible to develop an architecture that is able to integrate forensic data from the known and not known internal data structure of system generated files by the Windows Operating System; display the content information of those files in the form of narrative constructs and in a graphical manner; and to implement a proof of concept prototype tool capable of visualising the Event logs and Swap files, many additional areas that need to be resolved were identified. During the implementation stage of this project, a number of issues were revealed that could not initially be addressed. Some of the issues identified are improvements that can be made to the SAFTool in its current state; others are possible feature improvements that could make the SAFTool more feature-rich. These are discussed below:

- There are some speed issues that should be investigated. The process of analysing Swap files can take some time when large file sizes are being processed. However, the architecture works and given further research, the speed issues can be identified and addressed.
- The architecture has been created as a proof of concept prototype tool and not meant to be distributed, but through the experiments conducted it seems that its applications are stable and functional. The tool could be used in the academic environment as a teaching aid allowing others to examine the work and perhaps add their own contributions to the project.
- As the SAFTool is able to parse different data structure formats of system artefacts, it could be further developed to recognise and present other versions of the Windows operating system and all the various types of computer operating systems, both proprietary and open source.
- SAFTool is not necessary confined purely to system artefacts analysis. As a data examining architecture, it can be applied to different examining paradigms. For example, be able to recognise and present a wide range of files data structure, if any in the future and incorporating

recovering internal structure of processes from the memory (Craigier et al., 2005; Sutherland et al., 2008). Further, additional spectrum of data container could be used. Investigators have an increasing need to share digital evidence between different organisations and analysis tools. But today's investigators are hindered by a variety of independently developed and incompatible formats used to store digital evidence (EnCase, 2005; Garfinkel, 2006; The Common Digital Evidence Storage Format Working Group, 2006; Turner, 2006; and Pladna, 2008).

- To provide the investigator with an additional resource the system could also include a flat text file with an explanation / interpretation of the structure of the artefact. This can then be used as a reference for the investigator or to add an interpretation of material for a court report.
- As the SAFTool is able to be used in post-mortem analysis, there is a need to view running processes in conducting live investigations. In addition to physical media, investigators may also obtain evidence from other sources such as live memory analysis. Where the capture of live memory is not possible, some evidence of memory activity can be recovered from the Swap files left on the hard drive. The evolution of technology has a need for live analysis than a post-mortem one. With such live investigation, the forensic analysis system will more flexible as it can be modified to visualise RAM content when analysing memory.

## BIBLIOGRAPHY

AccessData (2009) *AccessData BootCamp Training Manual*. AccessData Corporation, Lindon, U.S.A.

ACPO (2003) *Good Practice Guide for Computer-Based Electronic Evidence*. United Kingdom Association of Chief Police Officers. Available at: [http://www.7safe.com/electronic\\_evidence/ACPO\\_guidelines\\_computer\\_evidence.pdf](http://www.7safe.com/electronic_evidence/ACPO_guidelines_computer_evidence.pdf) (Accessed: 14 July 2008).

Adamson, I. T. (1996) *Data Structures and Algorithms: A First Course*. Springer.

Adobe Systems Incorporated (2011) HomeSite. Available at: <http://rup.hops-http://www.adobe.com/products/homesite/> (Accessed: 14 February 2011).

Adobe Systems Incorporated (2011) DreamWeaver. Available at: <http://www.adobe.com/uk/products/dreamweaver/?promoid=BPCVH> (Accessed: 14 February 2011).

Alink, W., Bhoedjang, R. A. F., Boncz, P. A. and Vries, A. P. (2006) 'XIRAF – XML-based Indexing And Querying For Digital Forensics', *Journal of Digital Investigation*, Volume 3, Supplement 1, pp. 50-58.

Allen, R. (2005) *Windows Server Cookbook*. Available at: <http://my.safaribooksonline.com/0596006330/windowsvrckbk-APP-F?portal=o-reilly>. (Accessed: 6 January 2011).

Ambler, S. W. (2003) *UML 2 Component Diagram, Agile Modeling*. Available at: <http://www.agilemodeling.com/artifacts/component/Diagram.htm>. (Accessed: 12 June 2010).

Anderson, M. R. (2005) *Electronic Fingerprints Computer Evidence Comes Of Age*. Available at: <http://www.forensics-intl.com/art2.html>. (Accessed: 3 December 2007).

Anson, S. and Bunting, S. (2007) *Mastering Windows Network Forensics and Investigation*. Indianapolis, Indiana: Wiley Publishing, Inc.

Arnott, D. (2006) 'Cognitive Biases and Decision Support Systems Development: A Design Science Approach', *Information Systems Journal*, January, pp. 55.

Arthur, K. K. and Venter, H. S. (2005) *An Investigation Into Computer Forensic Tools*. Available at: <http://www.forensicfocus.com/computer-forensic-tools-investigation> (Accessed: 14 July 2008).

Ashcroft, J. (2001) *Electronic Crime Scene Investigation: A Guide for First Responders*. Available at: <http://www.ncjrs.gov/pdffiles1/nij/187736.pdf>. (Accessed: 27 February 2008).

Ashcroft, J., Daniels, D. J. and Hart, S. V. (2004) *Forensic Examination of Digital Evidence: A Guide for Law Enforcement*. Available at: <http://www.ncjrs.gov/pdffiles1/nij/199408.pdf>. (Accessed: 16 June 2008).

Avison, D. and Fitzgerald, G. (2003) *Information Systems Development: Methodologies, Techniques and Tools*. New York: McGraw Hill Education.

Ayers, D. (2009) 'A Second Generation Computer Forensic Analysis System', *Journal of Digital Investigation*, Volume 6, Supplement 1, pp. 34-42.

Bangeman, E. (2006) *Microsoft doesn't like PCs sold without Windows*. Available at: <http://arstechnica.com/news.ars/post/20060405-6531.html>. (Accessed: 16 June 2008).

Bejtlich, R., Jones, K. and Rose, C. W. (2005) *Windows Live Response for Collecting and Analysing Forensically Sound Evidence*. Available at: <http://www.informit.com/articles/article.aspx?p=417509>. (Accessed: 25 July 2007).

Bennett, S., McRobb, S. and Farmer, R. (2006) *Object-Oriented Systems Analysis And Design Using UML*. London: McGraw Hill.

Bilic, N. (2006) *System Event Viewer Tips*. Available at: <http://technet.microsoft.com/en-us/library/aa996105.aspx> (Accessed: 4 June 2008).

BJA (2010) *Bureau of Justice Assistance Evaluation and Performance Measurement Glossary*. Available at: [http://www.ojp.usdoj.gov/BJA/evaluation/glossary/glossary\\_i.htm](http://www.ojp.usdoj.gov/BJA/evaluation/glossary/glossary_i.htm) (Accessed: 23 November 2010).

Bos, J. and Knijff, R. (2005) 'TULP2G - An Open Source Forensic Software Framework for Acquiring and Decoding Data Stored in Electronic Devices', *International Journal of Digital Evidence*, Fall 2005, Volume 4, Issue 2.

Boudreau, T., Glick, J., Greene, S., Spurlin, V. and Woehr, J. (2003) *NetBeans: The Definitive Guide*. California, United States of America: O'Reilly & Associates.

Bradley, T. (2005) *Windows security threat tips: security log data: Make the most of it*. Available at: [http://searchwindowssecurity.techtarget.com/tip/289483,sid45\\_gci1107594,00.html](http://searchwindowssecurity.techtarget.com/tip/289483,sid45_gci1107594,00.html) (Accessed: 13 July 2007).

Brandel, M. (2008) *Rules of Evidence - Digital Forensics Tools*. Available at: <http://www.csoonline.com/article/374763/rules-of-evidence-digital-forensics-tools> (Accessed: 14 July 2009).

Britz, M. (2004) *Computer Forensics and Cyber Crime: An Introduction*. Upper Saddle River, New Jersey: Pearson Prentice Hall.

Brown, R., Palm, B. and Vel, O. de (2005) *Design of a Digital Forensics Image Mining System*. Lecture Notes in Computer Science 2005. Number 3683, pages 395-404. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.7935>  
Available at: <http://www.springerlink.com/content/3a7t7cxk3mdrajb0/> (Accessed: 13 January 2010).

Bryson, C. and Stevens, S. (2002) *Tool Testing And Analytical Methodology, Handbook Of Computer Crime Investigation: Forensic Tools And Technology*. San Diego, California: Academic Press.

Bunting, S. and Wei, W. (2006) *EnCase Computer Forensics The Official EnCE: Encase Certified Examiner Study Guide*. Indianapolis, Indiana: Wiley Publishing.

Cadenhead, R. and Lemay, L. (2007) *Sams Teach Yourself Java 6 in 21 Days*. United States of America: Bronkella Publishing.



Caloyannides, M. A., (2001) *Computer Forensics and Privacy*. Boston: Artech House.

Carrier, B. (2002) *Open Source Digital Forensics Tools: The Legal Argument*. Available at: [http://www.packetstormsecurity.org/papers/IDS/atstake\\_opensource\\_forensics.pdf](http://www.packetstormsecurity.org/papers/IDS/atstake_opensource_forensics.pdf) (Accessed: 14 July 2008).

Carrier, B. (2003) 'Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers', *International Journal of Digital Evidence*, 1(4).

Carrier, B. and Spafford, E., H. (2003) 'Getting Physical with the Digital Investigation Process', *International Journal of Digital Evidence*, 2(2).

Carrier, B. (2005) *File System Forensic Analysis*. Upper Saddle River, NJ: Addison Wesley.

Carrier, B. D. (2006) *Basic Digital Forensic Investigation Concepts*. Available at: [http://www.digital-evidence.org/di\\_basics.html](http://www.digital-evidence.org/di_basics.html) (Accessed: 12 February 2011).

Carrier, B. (2010) *The Sleuth Kit (TSK) and Autopsy: Open Source Digital Investigation Tools*. Available at: <http://www.sleuthkit.org/> (Accessed: 12 February 2011).

Carvey, H. (2004) 'Instant Messaging Investigations On A Live Windows XP System', *Journal of Digital Investigation*, 1(4), pp. 256-260.

Carvey, H. and Altheide, C. (2005) 'Tracking USB Storage: Analysis of Windows Artifacts Generated by USB Storage Devices', *Journal of Digital Investigation*, 2(2), pp. 94-100.

Carvey, H. (2007) *Windows Forensic Analysis DVD Toolkit*. Burlington: Syngress Publishing.

Casey, E. (2000), *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, Academic Press

Casey, E. (2002) 'Practical Approaches to Recovering Encrypted Digital Evidence', *International Journal of Digital Evidence*, 1(3).

Casey, E. (2004) *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, Second Edition*. San Diego, California: Elsevier Academic Press.

Casey, E. and Larson, T. (2004) *Digital Evidence Examination Guidelines, Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, Second Edition*. San Diego, California: Elsevier Academic Press.

Casey, E. and Turvey, B. (2004) *Investigative Reconstruction With Digital Evidence, Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, Second Edition*. San Diego, California: Elsevier Academic Press.

Casey, E. (2010) *Handbook of Digital Forensic and Investigation*. London: Elsevier Inc.

CERT (2005) *First Responders Guide to Computer Forensics*. Carnegie Mellon Software Engineering Institute, United States of America. Available at: [www.cert.org/archive/pdf/FRGCF\\_v1.3.pdf](http://www.cert.org/archive/pdf/FRGCF_v1.3.pdf) (Accessed: 14 July 2008).

Chappell, G. (2010) *The INDEX.DAT File Format*. Available at: <http://www.geoffchappell.com/viewer.htm?doc=studies/windows/ie/wininet/api/urlcache/indexdat.htm> (Accessed: 11 January 2011).

Collins, W. (2008) *Collins English Dictionary*, HarperCollins Publishers, Glasgow.

Conti, G., Dean, E., Sinda, M. and Sangster, B. (2008) 'Visual Reverse Engineering Of Binary And Data Files'. In *Proceedings of the 5th International Workshop on Visualisation for Computer Security*. pp. 1-17. Available at: [http://www.rumint.org/gregconti/publications/2008\\_VizSEC\\_FileVisualization\\_v53\\_final.pdf](http://www.rumint.org/gregconti/publications/2008_VizSEC_FileVisualization_v53_final.pdf) (Accessed: 14 July 2008).

Craiger, J. P., Pollitt, M. and Swauger, J. (2005) *Law Enforcement and Digital Evidence*. Available at: <http://ncfs.org/craiger.delf.revision.pdf> (Accessed: 7 August 2007).

Cummings, R. and Lowry, J. (2003) *Computer Forensic 101 and Incident Response*. Available at: [http://isacala.org/doc/2003oct1\\_workshop\\_pres.pdf](http://isacala.org/doc/2003oct1_workshop_pres.pdf) (Accessed: 7 August 2007).

Daintith, J. (2004) *"Software Tool": A Dictionary of Computing*. Available at: <http://www.encyclopedia.com/doc/1011-softwaretool.html> (Accessed: 16 April 2010).

Davia, H. R. (2000) *Fraud 101: Techniques And Strategies For Detection*. New York: John Wiley & Sons, Inc.

Dell (2010) The Official Site for Computer and PCs, Dell UK. Available at: <http://www.dell.co.uk/> (Accessed: 14 July 2010).

Derek, B. (2007) *Software Development Glossary*. Department of Computer Science, University College Cork, Ireland. Available at: <http://www.cs.ucc.ie/~dgb/courses/swd/glossary.html> (Accessed: 23 November 2010).

Detwiler, B. (2008) Poll: Which of the Following Windows Versions is the Most Prevalent Among Your End Users? Available at: <http://blogs.techrepublic.com.com/itdojo/?p=147> (Accessed: 7 August 2008).

Dickson, M. (2006) 'An Examination Into MSN Messenger 7.5 Contact Identification', *Journal of Digital Investigation*, 3(2), pp. 79-83.

Digitivity (2008). The Evolution of Relational Database Management System. Digitivity Electrical & Electronics Industry MarketPlace. Available at: <http://www.digitivity.com/articles/2008/10/evolution-of-relational-database.html> (Accessed: 14 July 2010).

DoJ (2007) *Digital Forensic Analysis Methodology Flowchart*. Department of Justice, Computer Crime and Intellectual Property Section, United States of America. Available at: [http://www.cybercrime.gov/forensics\\_chart.pdf](http://www.cybercrime.gov/forensics_chart.pdf) (Accessed: 28 October 2010).

Dongen, W. S. V. (2007) 'Forensic Artefacts left by Windows Live Messenger 8.0', *Journal of Digital Investigation*, 4(2), pp. 73-87.

Elcomsoft (2005) *Advanced Registry Tracer*. Available at: <http://www.elcomsoft.com/art.html> (Accessed: 3 December 2007).

EPF Copyright (2010). Guideline: Example: Design Mechanisms. Available at: [http://epf.eclipse.org/wikis/abrd/core.tech.common.extend\\_supp/guidances/guidelines/example\\_design\\_mechanisms\\_7762C0FB.html](http://epf.eclipse.org/wikis/abrd/core.tech.common.extend_supp/guidances/guidelines/example_design_mechanisms_7762C0FB.html) (Accessed: 14 July 2010).

Finkelstein, A. (2000) *Software Engineering*. Available at: [http://eprints.ucl.ac.uk/1119/1/13.5\\_seencyc.pdf](http://eprints.ucl.ac.uk/1119/1/13.5_seencyc.pdf) (Accessed: 11 February 2011).

Fitzgerald, E. (2006) *HOWTO Understand the Microsoft Windows Event Log*. Available at: [http://www.splunkbase.com/howtos/Operating\\_Systems/Windows/howto:HOWTOunderstandMSEventLog](http://www.splunkbase.com/howtos/Operating_Systems/Windows/howto:HOWTOunderstandMSEventLog) (Accessed: 11 July 2007).

Frauenheim, E. (2004) *Storage Hardware Sales on the Rise*. CNET News. Available at: [http://news.cnet.com/2100-1015\\_3-5170267.html?tag=fd\\_nbs\\_ent](http://news.cnet.com/2100-1015_3-5170267.html?tag=fd_nbs_ent) (Accessed: 28 September 2010).

Free Internet Window Washer (2010) *Free Internet Window Washer. Eusing Software*. Available at: [http://www.eusing.com/Window\\_Washer/Window\\_Washer.htm](http://www.eusing.com/Window_Washer/Window_Washer.htm) (Accessed: 3 February 2011).

Fry, B. (2007) *Visualising Data*. Sebastopol, CA: O'Reilly Media, Inc.

Gagnon, L. (2008) *Data Forensics: Tools and Techniques for a Basic Examination*. Available at: [http://www.esentire.com/user\\_files/images/File/Data%20Forensics%20What%20You%20Need%20to%20Know.pdf](http://www.esentire.com/user_files/images/File/Data%20Forensics%20What%20You%20Need%20to%20Know.pdf) (Accessed: 12 August 2010).

Garfinkel, S. L. (2006) 'Forensic Feature Extraction and Cross-Drive Analysis', *Journal of Digital Investigation*, Volume 3, Supplement 1, pp. 71-81.

Geiger, M. and Cranor, L. F. (2006) 'Scrubbing Stubborn Data: An Evaluation of Counter-Forensic Privacy Tools', *Journal of Security & Privacy*, 4(5), pp. 16-25.

GFI (2007) *LANguard Security Event Log Monitor*. Available at: <http://www.gfi.com/> (Accessed: 3 December 2007).

Ghavalas, B. and Philips, A. (2005) 'Trojan defence: A Forensic View Part II', *Journal of Digital Investigation*, 2(2), pp. 133-136.

Gillam, Wm. B. and Rogers, M. (2005) 'File Hound: A Forensic Tool For First Responders'. In *Proceedings of the 5th Annual Digital Forensic Research Workshop, DFRWS 2005*. Available at: [http://www.dfrws.org/2005/proceedings/gillam\\_filehound.pdf](http://www.dfrws.org/2005/proceedings/gillam_filehound.pdf) (Accessed: 14 July 2008).

Gould, S. R. (2004) *Index Dat Spy*. Available at: [http://www.stevengould.org/index.php?option=com\\_content&task=view&id=47&Itemid=88](http://www.stevengould.org/index.php?option=com_content&task=view&id=47&Itemid=88) (Accessed: 12 January 2011).

Greene, T. C. (2007) *Clearing swap and hibernation files properly*. Available at: [http://www.theregister.co.uk/2007/05/05/wipe\\_swap\\_file/](http://www.theregister.co.uk/2007/05/05/wipe_swap_file/) (Accessed: 3 December 2007).

Grochowski, E. and Halem, R. D. (2003) *Technological Impact of Magnetic Hard Disk Drives on Storage Systems*. Available at: <http://www.cs.princeton.edu/courses/archive/spr05/cos598E/bib/grochowski.pdf> (Accessed: 19 August 2010).

Guidance Software (2005) *Encase Forensic Version 5 User Manual*. Available at: <http://www.guidancesoftware.com> (Accessed: 14 July 2008).

Guillermo, F., Trifas, M., Brown, D., Francia, R. and Scott, C. (2007) *Forensic Data Visualization System: Improving Security Through Automation: Computer Security Conference*. Myrtle Beach, SC, USA, 11-13 April.

Hailey, S. (2003) *What Is Computer Forensics*, CyberSecurity Institute. Available at: <http://www.cybersecurityinstitute.biz/forensics.htm> (Accessed: 14 July 2008).

Haiping, C., Delin, L. and Qinquan, G. (2009) *IE Internet Information Forensics Technology in Unallocated Disk Space: International Symposium On Computer Network and Multimedia Technology*.

Harms, K. (2006) 'Forensic Analysis of System Restore Points in Microsoft Windows XP', *Journal of Digital Investigation*, 3(3), pp. 151-158.

Harvey, P. (2004) *Open Source Security Tools: A Practical Guide to Security Applications*. Available at: <http://www.informit.com/store/product.aspx?isbn=0321194438> (Accessed: 12 February 2010).

Hashim, N. and Sutherland, I. (2007) *System Data Artefacts: Event Logs and Swap Files*. 2<sup>nd</sup> Research Student Workshop, University of Glamorgan, Cardiff, UK.

Hashim, N. and Sutherland, I. (2010) *An Architecture For The Forensic Analysis of Windows System Artefacts: 2<sup>nd</sup> International ICST Conference on Digital Forensics and Cyber Crime*. Abu Dhabi, UAE, 3-6 October.

Hautefeuille, B. (2011) A Simpler Way of Getting .NET Objects Out of ADO.NET. Available at: <http://www.15seconds.com/issue/031013.htm> (Accessed: 14 March 2011).

Hay, S. A. (2005) *Windows File Analyser Guidance*. Available at: <http://www.mitec.cz/Downloads/WFA%20Guidance.pdf> (Accessed: 14 July 2008).

Hitachi (2008), *The 4TB Hard Drive*. 2008 Hitachi Global Storage Technologies. Available at: [http://www.hitachigstwhatsnext.com/en/Future\\_Technology/4TB\\_hard\\_drive/](http://www.hitachigstwhatsnext.com/en/Future_Technology/4TB_hard_drive/) (Accessed: 28 September 2010).

Hosmer, C. (2002) *Time Lining Computer Evidence: Information Technology Conference 1998*.

Howell, B. A. (2005) 'Digital Forensics: Sleuthing on Hard Drives and Networks', *Journal of The Vermont Bar*. Available at: <http://www.strozfriedberg.com/files/Publication/884a031c-755c-40d8-a09c-2abaa57e9496/Presentation/Publication-Attachment/28e367df-8658-4fe9-a4c0-26011c249931/VTBDigitalForensicsArticle.pdf> (Accessed: 12 February 2010).

Howlett, T. (2004) *Open Source Security Tools: A Practical Guide to Security Applications*. Available at: <http://www.informit.com/store/product.aspx?isbn=0321194438> (Accessed: 12 February 2010).

IBM (2006) *DB2 Database for Linux, Unix and Windows*. International Business Machines. Available at: <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0004100.htm> (Accessed: 14 July 2008).

IBM (2011a). Unified Modeling Language. IBM Rational Software. Available at: <http://www-01.ibm.com/software/rational/uml/> (Accessed: 14 February 2011).

IBM (2011b). Informix product family. IBM Information Management. Available at: <http://www-01.ibm.com/software/data/informix/> (Accessed: 14 February 2011).

Ieong, R. S. C. (2006) 'FORZA - Digital Forensics Investigation Framework That Incorporate Legal Issues', *Journal of Digital Investigation*, Volume 3, Supplement 1, pp. 29-36.



International Telecommunication Union (2009) *Proposal of New Work Item on Digital Evidence Exchange File Format*. Telecommunication Standardisation Sector Republic of Korea .

Jeffrey, R. and Clark, J. D. (2000) *Programming Server Side Applications for Microsoft Windows 2000*. Available at: [http://www.zanshu.com/ebook/175\\_13server/HTML/](http://www.zanshu.com/ebook/175_13server/HTML/) (Accessed: 7 August 2007).

Jones, A., Valli, C., Sutherland, I. and Thomas P. (2006) *An Analysis of Information Remaining on Disks Offered for Sale on the Second Hand Market*. Journal of Digital Forensics, Security and Law. Volume 1, Issue 3.

Jones, A., Valli, C. and Sutherland, I. (2008) *Analysis of Information Remaining on Hand Held Devices Offered for Sale on the Second Hand Market*. Journal of Digital Forensics, Security and Law. Volume 3, Issue 2.

Jones, A., Dardick G., Davies G., Sutherland, I. and Valli, C. (2009) *The 2008 Analysis of Information Remaining on Disks Offered for Sale on the Second Hand Market*. Journal of International Commercial Law and Technology, Vol.4 (3) 2009.

Jones, K. (2003) Visual Computer Forensic Analysis: Galleta, <http://www.law.com/jsp/lawtechnologynews/PubArticleLTN.jsp?id=120242824-8638> (Accessed: 7 August 2007).

Kahvedzic, D. and Kechadi, T. (2008) *Extraction of User Activity Through Comparison of Windows Restore Point: 6th Australian Digital Forensics Conference 2008*.

Kahvedzic, D. and Kechadi, T. (2010) *Extraction of User Activity Through Comparison of Windows Restore Point: 2<sup>nd</sup> International ICST Conference on Digital Forensics and Cyber Crime*. Abu Dhabi, UAE, 3-6 October.

Kale, K. V. (2007) *Investigative System for Evidences Collection in Internet Explorer Cache Files: Advances in Computer Vision and Information Technology*. Available at: [http://books.google.co.uk/books?id=pNKxKYHL2RYC&pg=PA143&lpg=PA143&dq=HASH+of+index.dat&source=bl&ots=mNvNM8Ah9g&sig=2GUcYc5mvbomGw\\_ozKCdtIrjJY&hl=en&ei=sj0rTbLcFp2ShAeDsZ2IAg&sa=X&oi=book\\_result&ct=result&resnum=9&sqi=2&ved=0CFIQ6AEwCA#v=onepage&q=HASH%20of%20index.dat&f=false](http://books.google.co.uk/books?id=pNKxKYHL2RYC&pg=PA143&lpg=PA143&dq=HASH+of+index.dat&source=bl&ots=mNvNM8Ah9g&sig=2GUcYc5mvbomGw_ozKCdtIrjJY&hl=en&ei=sj0rTbLcFp2ShAeDsZ2IAg&sa=X&oi=book_result&ct=result&resnum=9&sqi=2&ved=0CFIQ6AEwCA#v=onepage&q=HASH%20of%20index.dat&f=false) (Accessed: 25 February 2010).

Kavanagh, P. (2004) *Open Source Software: Implementation and Management*. Available at: [http://books.google.co.uk/books?id=CHkHNChvPqIC&printsec=frontcover&dq=open+source+software&source=bl&ots=iYLLDfZIYz&sig=bZR UeLlJA\\_AUM5lo46krTeLQiw&hl=en&ei=HwYrTcSUA5CC5Ab9u9iZCg&sa=X&oi=book\\_result&ct=result&resnum=9&ved=0CFcQ6AEwCDgK#v=onepage&q&f=false](http://books.google.co.uk/books?id=CHkHNChvPqIC&printsec=frontcover&dq=open+source+software&source=bl&ots=iYLLDfZIYz&sig=bZR UeLlJA_AUM5lo46krTeLQiw&hl=en&ei=HwYrTcSUA5CC5Ab9u9iZCg&sa=X&oi=book_result&ct=result&resnum=9&ved=0CFcQ6AEwCDgK#v=onepage&q&f=false) (Accessed: 25 February 2010).

KayKeys (2005) *Object-Oriented Design/Process Concepts*. Available at: <http://kaykeys.net/science/computerwork/oodesign/index.html> (Accessed: 14 July 2008).

Keizer, G. (2008) *Leopard drubs Vista in corporate satisfaction survey, Computerworld Operating Systems*. Available at: <http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9072218> (Accessed: 14 July 2008).

Kerlinger, F. N. (1986) *Foundations of Behavioral Research* (3rd edn.). New York: Holt, Rinehart and Winston.

Kuhn, T. (1996) *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press.

Kleber, R. and Galvao, M. (2006) 'Computer Forensics with The Sleuth Kit and The Autopsy Forensic Browser', *The International Journal of Forensic Computer Science*, 1, pp. 41-44.

Knuth, D. E. (1997) *Art of Computer Programming, Volume 1: Fundamental Algorithms (3<sup>rd</sup> Edition)*. Addison-Wesley Professional.

Kornblum, J. (2007) 'Using Every Part of the Buffalo in Windows Memory Analysis', *Journal of Digital Investigation*, 4(1), pp. 24-29.

Kruse II, W. G. and Heiser, J. G. (2002) *Computer Forensics: Incident Response Essentials*. Indianapolis: Addison-Wesley.

Kumar, R. (2005) *Research Methodology (2<sup>nd</sup> edn.)*. London: SAGE Publications Ltd.

La Bella, R. (2004) *Know Your Enemy: Learning About Security Threats/The Honeynet Project – 2<sup>nd</sup> edn.*. Boston: Pearson Education, Inc.,

Lakatos, I. (1978) *The Methodology of Scientific Research Programmes (John Worral and Gregory Currie, Eds.)*. Cambridge: Cambridge University Press.

Lee, S., Savoldi, A., Lee, S., Lim, J. (2007a) 'Windows Pagefile Collection and Analysis for a Live Forensics Context', *Journal of Future Generation Communication And Networking*. FGCN 2007. 6-8 Dec. Pages 97-101. The British Library.

Lee, S., Savoldi, A., Lee, S. and Lim, J. (2007b) *Password Recovery Using An Evidence Collection Tool And Countermeasures: 3<sup>rd</sup> International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Volume II, Pages 97-102. The British Library.

Leedy, P. D. and Ormrod, J. E. (2005) *Practical Research Planning and Design, 8th Edition*. New Jersey: Pearson Merrill Prentice Hall.

Lewis, J. A. (2004) *Where Data Resides – Data Discovery from the Inside Out*. Available at: [http://www.digitalmountain.com/recent\\_article\\_3](http://www.digitalmountain.com/recent_article_3) (Accessed: 3 December 2007).

Love, T. (2000) *Theoretical Perspectives, Design Research and the PhD Thesis. In Doctoral Education in Design, Foundations for the Future, edited by Durling, D. and Friedman, K.* Staffordshire, UK: Staffordshire University Press.

Luo, V. C. (2007) *Tracing USB Device Artefacts on Windows XP Operating System for Forensic Purpose*. Available at: [http://scissec.scis.ecu.edu.au/publications/2007/forensics/23\\_Luo\\_Tracing\\_USB\\_Device\\_artefacts\\_on\\_Windows\\_XP.pdf](http://scissec.scis.ecu.edu.au/publications/2007/forensics/23_Luo_Tracing_USB_Device_artefacts_on_Windows_XP.pdf) (Accessed: 14 July 2008).

Lyle, J. R. (2006) *The Contribution of Tool Testing to the Challenge of Responding to an IT Adversary*. Available at: [http://www1.giev.de/fachbereiche/sicherheit/fg/sidar/imf/imf2006/21\\_Lyle\\_imf-stuttgart-06.pdf](http://www1.giev.de/fachbereiche/sicherheit/fg/sidar/imf/imf2006/21_Lyle_imf-stuttgart-06.pdf) (Accessed: 13 July 2009).

MacVittie, L. (2010) Following Google's Lead on Security? Don't Forget to Encrypt Cookies. f5 DevCentral. Available at: <http://devcentral.f5.com/weblogs/macvittie/archive/2010/01/15/google-gmail-ssl-cookie-encryption.aspx> (Accessed: 13 January 2011).

Mandia, K., Proise, C. and Pepe, M. (2003) *Incident Response & Computer Forensics, Second Edition*. California: McGraw-Hill/Osborne.

Mandiant (2006) *Web Historian*. MANDIANT Corporation, Available at: [http://www.mandiant.com/products/free\\_software/web\\_historian/](http://www.mandiant.com/products/free_software/web_historian/) (Accessed: 12 January 2011).

Manson, D., Carlin, A., Ramos, S., Gyger, A., Kaufman, M. and Treichelt, J. (2007) *Is the Open Way a Better Way? Digital Forensics Using Open Source Tools*. Available at: <http://www.computer.org/portal/web/csdl/doi/10.1109/HICSS.2007.301> (Accessed: 12 February 2010).

Marcella, A. J. and Greenfield, R. S. (2002) *Cyber Forensics: A Field Manual For Collecting, Examining and Preserving Evidence of Computer Crimes*. Boca Raton, Florida: Auerbach Publications.

Marcella, A. J. and Mendenez, D. (2008) *Cyber Forensics: A Field Manual For Collecting, Examining and Preserving Evidence of Computer Crimes*, Second Edition, Boca Raton, Florida, Auerbach Publications.

March, S. and Smith, G. (1995) 'Design and Natural Science Research on Information Technology'. *Journal of Decision Support Systems*, Volume 15 Issue 4, pp. 251-266.

Martin, R. C. (1995) *Designing Object Oriented C++ Applications Using the Booch Method*. Upper Saddle River, New Jersey: Prentice Hall Publications.

Mee, V., Tryfonas, T. and Sutherland, I. (2006) 'The Windows Registry As A Forensic Artefact: Illustrating Evidence Collection for Internet Usage', *Journal of Digital Investigation*, 3(3), pp. 166-173.

Mee, V. (2009) *The Application of Visualisation Architecture to the Windows Registry as a Forensic Object to Aid in the Forensic Process: PhD Thesis*. University of Glamorgan. pp. 127-132.

Microsoft Computer Dictionary (2002), Microsoft Computer Dictionary: Fifth Edition. Microsoft Press

Microsoft (2005) *Log Parser 2.2*. Microsoft Technet, Available at: <http://www.microsoft.com/technet/scriptcenter/tools/logparser/default.aspx> (Accessed: 14 July 2008).

Microsoft (2005a) *Chapter 9 - Monitoring Events*. Windows NT Server Product Documentation, Available at: <http://www.microsoft.com/resources/documentation/windowsnt/4/server/proddocs/enus/concept/xcp09.aspx?mfr=true> (Accessed: 6 January 2011).

Microsoft Corporation (2005) What is JView. Microsoft Security Bulletin MS05-037. Available at: <http://www.microsoft.com/technet/security/Bulletin/MS05-037.aspx> (Accessed: 18 November 2010).

Microsoft (2007) *INFO: Working with the FILETIME Structure*. Microsoft Help and Support. Available at: <http://support.microsoft.com/kb/188768> (Accessed: 14 July 2008).

Microsoft Corporation (2011). FrontPage. Available at: <http://office.microsoft.com/en-us/frontpage-help/> (Accessed: 14 January 2011).

Microsoft Corporation (2011a). Overview of the JDBC Driver. Available at: <http://msdn.microsoft.com/en-us/library/ms378749.aspx> (Accessed: 14 January 2011).

Microsoft TechNet (2007) *Fundamental Computer Investigation Guide For Windows: Overview*. Available at: [http://www.microsoft.com/downloads/details.aspx?FamilyId=71B986EC-B3F1-4C14AC70EC0EB8ED9D57& displaylang=en](http://www.microsoft.com/downloads/details.aspx?FamilyId=71B986EC-B3F1-4C14AC70EC0EB8ED9D57&displaylang=en) (Accessed: 14 July 2008).

Middleton, B. (2002) *Cyber Crime Investigator's Field Guide*. Florida: CRC Press LLC.

Mil Incorporated (2010) *What is in the Index.dat files?*. Available at: [http://www.milincorporated.com/a3\\_index.dat.html](http://www.milincorporated.com/a3_index.dat.html) (Accessed: 11 January 2011).

Metz, J. (2009) *MSIE Cache File (index.dat) format specification*. Available at: [http://mirror.transact.net.au/sourceforge/l/project/li/libmsiecf/Documentation/MSIE%20Cache%20File%20format/MSIE%20Cache%20File%20\(index.dat\)%20format.pdf](http://mirror.transact.net.au/sourceforge/l/project/li/libmsiecf/Documentation/MSIE%20Cache%20File%20format/MSIE%20Cache%20File%20(index.dat)%20format.pdf) (Accessed: 11 January 2011).

Mocas, S. (2004) 'Building Theoretical Underpinnings for Digital Forensics Research', *Journal of Digital Investigation*, 1(1), pp. 61-68.

Mohay, G., Anderson, A., Collie, B., De Vel, O. and McKemmish, R. (2003) *Computer and Intrusion Forensic*. Norwood: Artech House, Inc.

Morgan, T., D. (2009) *The Windows NT Registry File Format Version 0.4*. Sentinel Chicken Networks. Available at: [http://www.sentinelchicken.com/research/registry\\_format](http://www.sentinelchicken.com/research/registry_format) (Accessed: 8 February 2010).

Morris, J. (2003) *Forensics on the Windows Platform, Part One and Two*. Available at: <http://www.securityfocus.com/print/infocus/1661> (Accessed: 30 April 2007).

Murphey, R. (2007) 'Automated Windows Event Log Forensics', *Journal of Digital Investigation*, Volume 4, Supplement1, pp. 92-100.

Murr, M. (2006) *The Basics of How Digital Forensics Tools Work* 3 December, 2006. Forensic Computing Blog. Available at: <http://www.forensicblog.org/2006/2/03/the-basics-of-how-digital-forensics-tools-work/> (Accessed: 29 September 2008).

Murr, M. (2007) *How Digital Forensics Relates to Computing* 25 January, 2007. Forensic Computing Blog. Available at: <http://www.forensicblog.org/2007/01/25/how-digital-forensics-relates-to-computing/> (Accessed: 29 September 2008).

Murr, M. (2007) *How Forensics Tools Recover Digital Evidence (Data Structure)* 5 May, 2007. Forensic Computing Blog. Available at: <http://www.forensicblog.org/2007/05/05/how-forensic-tools-recover-digital-evidence-data-structures/> (Accessed: 29 September 2008).

Murr, M. (2007), *The Five Phases of Recovering Digital Evidence* 8 May, 2007. Forensic Computing Blog. Available at: <http://www.forensicblog.org/2007/05/08/the-five-phases-of-recovering-digital-evidence/> (Accessed: 29 September 2008).

Murr, M. (2009) *The Meaning of LEAK Records*. Forensic Computing Blog. Available at: <http://www.forensicblog.org/2009/09/10/the-meaning-of-leak-records/> (Accessed: 11 January 2011).

MyCERT (2007) *Fortinet Announces Top Reported Threats for November 2007, E-Security, Cyber Security Malaysia*, Volume 13 (Q4/2007).

Nelson, B., Phillips A., Enfinger, F. and Steuart, C. K. (2004) *Guide To Computer Forensics And Investigations*, Course Technology a Division of Thomson Technology, Canada.



Net Applications, (2009) Operating system market share. Market Share, Available at: <http://marketshare.hitslink.com/report.aspx?qprid=10&qpmr=15&qpdt=1&qpct=3&qpcal=1&qptimeframe=M&qpsp=122> (Accessed: 14 July 2010).

Newman, R. C. (2007) *Computer Forensics: Evidence Collection and Management*. Boca Raton: Auerbach Publications.

Nikkel, B. (2005) Digital Forensics using Linux and Open Source Tools. Available at: <http://www.digitalforensics.ch/nikkel05b.pdf> (Accessed: 14 July 2008).

Nolan, R., O'Sullivan, C., Branson, J. and Waits, C. (2005) *First Responders Guide to Computer Forensics*. Available at: [http://www.cert.org/archive/pdf/FRGCF\\_v1.3.pdf](http://www.cert.org/archive/pdf/FRGCF_v1.3.pdf) (Accessed: 14 July 2008).

Nourie, D. (2005) *Getting Started with an Integrated Development Environment (IDE)*. Available at: <http://java.sun.com/developer/technicalArticles/tools/-intro.html> (Accessed: 8 February 2011).

Object Management Group (2005) *UML Resource Page*. Available at: <http://uml-directory.omg.org/> (Accessed: 14 July 2008).

O'Conner, J. (2007) *Creating Extensible Applications With the Java Platform*. Available at: <http://java.sun.com/developer/technicalArticles/javase/extensible/index.html> (Accessed: 25 February 2010).

O'Leary, Z. (2005) *Researching Real-World Problems*. London, California, New Delhi, Singapore: SAGE Publications.

Okolica, J. and Peterson, G. L. (2010) 'Windows Operating Systems Agnostic Memory Analysis', *Journal of Digital Investigation*, Volume 7, Supplement 1, pp. 48-56.

Oracle (2011). Hardware and Software, Engineered to Work Together. Available at: <http://www.oracle.com/index.html> (Accessed: 14 January 2011).

OSI (2010) *Open Source Initiative*. Available at: <http://www.opensource.org/> (Accessed: 25 February 2010).

Petroni, N. L., Walters, A., Fraser, T. and Arbaugh, W. A. (2006) 'FATKit: A Framework for the Extraction and Analysis of Digital Forensic Data from Volatile System Memory', *Journal of Digital Investigation*, 3(4), pp. 197-210.

Pidanick, R. (2004) *An Investigation of Computer Forensics*. Available at: <http://www.isaca.org/Template.cfm?Section=Home&CONTENTID=19743&TEMPLATE=/ContentManagement/ContentDisplay.cfm> (Accessed: 14 July 2008).

Pladna, B. (2008) *Computer Forensics Procedures, Tools, and Digital Evidence Bags: What They Are and Who Should Use Them*. Available at: [http://www.infosecwriters.com/text\\_resources/pdf/BPladna\\_Computer\\_Forensic\\_Procedures.pdf](http://www.infosecwriters.com/text_resources/pdf/BPladna_Computer_Forensic_Procedures.pdf) (Accessed: 11 August 2008).

Poggenpohl, S. and Sato, K. (2003) 'Models of Dissertation Research in Design', *3rd Doctoral Education in Design Conference. Tsukuba, Japan*. October. Available at: [http://www.id.iit.edu/141/documents/tsukuba\\_2003.pdf](http://www.id.iit.edu/141/documents/tsukuba_2003.pdf) (Accessed: 11 August 2008).

POST (2005) *Open Source Software*. Available at: <http://www.parliament.uk/documents/post/postpn242.pdf> (Accessed: 25 February 2010).

Prentice Hall (2010) *Online Glossary*. Available at: [http://www.prenhall.com/rm\\_student/html/glossary/i\\_gloss.html](http://www.prenhall.com/rm_student/html/glossary/i_gloss.html) (Accessed: 23 November 2010).

ProDiscover (2003) *ProDiscover Computer Forensics*. Technology Pathways. Available at: <http://www.techpathways.com/DesktopDefault.aspx> (Accessed: 23 November 2010).

Purao, S. (2002) 'Design Research in the Technology of Information Systems: Truth or Dare'. In *GSU Department of CIS Working Paper*, Atlanta.

Rational Software Corporation (2003) Activity: Identify Design Elements. Available at: [http://rup.hops-fp6.org/process/activity/ac\\_iddes.htm](http://rup.hops-fp6.org/process/activity/ac_iddes.htm) (Accessed: 14 July 2010).

Read, H. (2009) *Data Exchange for Visualising Security Events (DEViSE): PhD Thesis*. University of Glamorgan. pp. 43-46.

Read, H., Blyth, A. and Sutherland, I. (2009) *A Unified Approach to Network Traffic and Network Security Visualisation: IEEE International Conference on Communication*. pp. 1-6.

Reyes, A., O'Shea, K., Steele, J., Hansen, J. R., Jean, B. R., Ralph, T. and Cunningham, B. (2007) *Cyber Crime Investigations: Bridging the Gaps Between, Security Professionals, Law Enforcement and Prosecutors*. Rockland, MA: Syngress Publishing, Inc.

Richard, G. G. and Roussev V. (2005), Scalpel: A Frugal, High Performance File Carver. In *Proceedings of the 2005 Digital Forensics Research Workshop*. New Orleans, L.A.

Richard, G. G. and Roussev V. (2006), 'Next Generation Digital Forensics'. *Communications of the ACM*, Volume 49, No. 2.

RIPA (2000) *The Regulation of Investigatory Powers Act 2000*. Available at: <http://www.yourprivacy.co.uk/what-ripa.html> (Accessed: 02 November 2010).

Rossi, M. and Sein, M. (2003) 'Design Research Workshop: A Proactive Research Approach'. *Presentation Delivered at IRIS26*, August. <http://www.cis.gsu.edu/~emonod/epistemology/Sein%20and%20Rossi%20-%20design%20research%20-%20IRIS.pdf> (Accessed: 11 November 2010).

Roussev, V. and Richard, G. G. (2004), Breaking the Performance Wall: the Case for Distributed Digital Forensics. In *Proceedings of the 2004 Digital Forensics Research Workshop*. Baltimore, MD.

Rubin, S. and Howell, B. A. (2006) *What You Need To Know About Digital Forensics*. Available at: <http://www.strozfriedberg.com/services/xprServiceDetailSF.aspx?xpST=ServiceDetail&service=5&op=Pubs> (Accessed: 11 August 2008).

Ruff, N. (2007) 'Windows Memory Forensics', *Journal in Computer Virology*. 4S, S92-S100.

Russinovich, M. (1999) *Inside the Registry*, Microsoft NT Magazine. Available at: <http://technet.microsoft.com/en-us/library/cc750583.aspx> (Accessed: 18 October 2007).

Rutman, L. (ed.) (1977) *Evaluation Research Methods: A Basic Guide*. Beverly Hills: Sage Publications.

Sanderson, B. (2004) *RAM, Virtual Memory, Pagefile and All That Stuff*, Microsoft Help and Support. Available at: <http://support.microsoft.com/default.aspx?scid=kb;en-us;555223> (Accessed: 18 October 2007).

Sansurooah, K. (2006) *Taxonomy Of Computer Forensics Methodologies and Procedures For Digital Evidence Seizure: 4th Australian Digital Forensics Conference*. Paper 32.

SAS (2010) *Concepts of Experimental Design*. Design Institute for Six Sigma Available at: <http://support.sas.com/resources/papers/sixsigma1.pdf>. (Accessed: 24 November 2010).

Schneider, G. M. and Bruell, S. C. (1998) *Concepts in Data Structures & Software Development: A Text for the Second Course in Computer Science*. Minnesota: West Publishing.

Schuster, A. (2006) 'Searching For Processes and Threats In Microsoft Windows Memory Dump', *Journal of Digital Investigation*, Volume 3, Supplement 1, pp. 10-16.

Schuster, A. (2007) 'Introducing the Microsoft Vista Event Log File Format', *Journal of Digital Investigation*, Volume 4, Supplement 1, pp. 65-72.

Schweitzer, D. (2003) *Incident Response: Computer Forensics Toolkit*. Indianapolis, Indiana: Wiley Publishing, Inc.

Shinder, T. (2002) *Take Control Of The Windows XP Pagefile*. Available at: <http://articles.techrepublic.com.com/5100-6346-1056269.html> (Accessed: 25 September 2007).

Shinder, D. L. and Tittel, E. (ed.) (2002) *Scene of the Cybercrime: Computer Forensics Handbook*. Rockland, MA: Syngress Publishing Inc.

Shnitman, A. (2000) *Unified Modeling Language*. SearchSoftwareQuality.com. Available at: <http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language> (Accessed: 14 July 2008).

Sheldon, B. (2002) *Forensic Analysis of Windows Systems, Handbook of Computer Crime Investigation*. London: Academic Press.

Simon, M. and Slay, J. (2010) *Recovery of Skype Application Activity Data from Physical Memory: 2010 International Conference on Availability, Reliability and Security*, pp.283-288,

Solomon, M. G., Barrett, D. and Broom, N. (2005) *Computer Forensic Jump Start*. California: SYBEX Inc.,

Sommer, P. (1998) *Digital Footprints: Assessing Computer Evidence*, Criminal Law Review Special Edition December 1998, pages 61-78. Available at: <http://www.pmsommer.com/CrimLR01.PDF> (Accessed: 14 July 2010).

Stacy, H. and Lunsford, P. (2006) *Computer Forensics For Law Enforcement*. Available at: [http://www.infosecwriters.com/text\\_resources/pdf/Forensics\\_HStacy.pdf](http://www.infosecwriters.com/text_resources/pdf/Forensics_HStacy.pdf) (Accessed: 14 July 2008).

Stair, R. M. and Reynolds, G. W. (2008) *Fundamentals of Information Systems, A Managerial Approach, Fourth Edition*. Massachusetts: Thomson Course Technology.

Sun Developer Network (2010) *About The Java Technology*. Available at: <http://java.sun.com/docs/books/tutorial/> (Accessed: 14 July 2010).

Sun Microsystems (2007) *The Benefits of Modular Programming*. Available at: [http://netbeans.org/project\\_downloads/usersguide/rcp-book-ch2.pdf](http://netbeans.org/project_downloads/usersguide/rcp-book-ch2.pdf) (Accessed: 14 July 2008).

Sutherland I., Evans, J., Tryfonas, T. and Blyth, A. J. C., (2008) 'Acquiring Volatile Operating System Data: Tools and Techniques'. *ACM SIGOPS Operating Systems Review*, 42(3).

Sutherland I., Davies, G. and Blyth, A. J. C., (2010) 'Malware and Steganography in Hard Disk Firmware'. *Journal of Computer Virology* 2010, DOI: 10.1007/s11416-010-0149-x. Online 10<sup>th</sup> January 2011

Svensson, A. (2005) *Computer Forensics Applied to Windows NTFS Computers*, Master Thesis Stockholm's University / Royal Institute of Technology Kista, Stockholm, Sweden. Available at: <http://www.dsv.su.se/en/seclab/pages/pdf-files/2005-x-268.pdf> (Accessed: 13 August 2007).

SWGDE (2006) *Best Practices for Computer Forensics*. Scientific Working Group on Digital Evidence, United State. Available at: [http://www.swgde.org/documents/wgde2006/Best\\_Practices\\_for\\_Computer\\_Forensics%20July06.pdf](http://www.swgde.org/documents/wgde2006/Best_Practices_for_Computer_Forensics%20July06.pdf) (Accessed: 14 July 2008).

Sybase (2011). Database Management. Available at: <http://www.sybase.com/products/databasemanagement> (Accessed: 14 January 2011).

Systemance (2006) *Index.dat Analyser*. Systemance Software Solutions. Available at: <http://www.systemance.com/indexdat.php> (Accessed: 12 January 2011).

TIBCO (2011) *Spotfire*. Available at: <http://spotfire.tibco.com/> (Accessed: 24 January 2011).

TCT (1999) *The Coroner's Toolkit*. Available at: <http://www.porcupine.org/forensics/tct.html> (Accessed: 23 November 2010).

Teerlink, S. and Erbacher, R. F. (2006) 'Foundations for Visual Forensic Analysis', In *Proceedings of the 7th IEEE Workshop on Information Assurance*, U. S. Military Academy, West Point.

The Common Digital Evidence Storage Format Working Group (2006) *Standardising Digital Evidence Storage*, Communication of the ACM.

The NT Insider (1998) *Windows NT Virtual Memory*, Available at: <http://www.osronline.com/article.cfm?article=71> (Accessed: 23 November 2008)

Thomas, L. K. (2003) *Reverse Engineering Index.dat*. Available at: <http://www.latenighthacking.com/projects/2003/reIndexDat/>. (Accessed: 9 December 2010).

TSK (2003) *The Sleuth Kit*. Available at: <http://www.sleuthkit.org/index.php> (Accessed: 23 November 2010).

Turnbull, B., Blundell, B. and Slay, J. (2006) 'Google Desktop As A Source Of Digital Evidence', *International Journal of Digital Evidence*, 5(1).

Turner, P. (2005) 'Unification of Digital Evidence From Disparate Sources (Digital Evidence Bags)', *Journal of Digital Investigation*. 2(3), pp. 223-228.

Turner, P. (2006) 'Selective and Intelligent Imaging Using Digital Evidence Bags', *Journal of Digital Investigation*, Volume 3, Supplement 1, pp. 59-64.



USDOJ (2001) *Electronic Crime Scene Investigation – A Guide for First Responders*. National Institute of Justice, U.S. Department of Justice. Available at: <http://www.ncjrs.gov/pdffiles1/nij/219941.pdf>. (Accessed: 14 July 2008).

University of Glamorgan (2008) *Unpublished Research Data*. Information Security Research Group, Faculty of Advanced Technology, University of Glamorgan. Pontypridd. United Kingdom.

Vacca, J. R. (2002) *Computer Forensics: Computer Crime Scene Investigation*. Hingham, Mass: Charles River Media.

Vaishnavi, V. and Kuechler, W. (2004) *Design Research in Information Systems*. Association for Information Systems. Available at: <http://ais.affiniscape.com/displaycommon.cfm?an=1&subarticlenbr=279>. (Accessed: 19 March 2009).

VisiData Limited (2011) *Sawmill*. Available at: <http://www.sawmill.co.uk/> (Accessed: 24 January 2011).

Vlastos, E. and Patel, A. (2007) 'An Open Source Forensic Tool To Visualise Digital Evidence', *International Journal of Computer Standards & Interfaces*. 30(1-2), pp. 8-19.

Volonino, L., Anzaldua, R. and Godwin, J. (2007) *Computer Forensics: Principles and Practices*. Upper Saddle River, New Jersey: Pearson Prentice Hall.

Vyavhare, A. (2009) Top Cyber Forensic Tools. Available at: <http://www.articleswave.com/computer-articles/top-cyber-forensic-tools.html>. (Accessed: 23 June 2009).

Walden, I. (2007) *Computer Crimes And Digital Investigations*. Oxford: University Press.

Wang, W. and Daniels, T. E. (2006) 'Building Evidence Graphs For Network Forensics Analysis', Computer Security Applications Conference, 21<sup>st</sup> Annual. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.2142&rep=rep1&type=pdf> (Accessed: 14 July 2008).

Whitehead, A. (2010) File Recovery from Recycle Bin. Available at: <http://free-backup.info/file-recovery-from-recycle-bin.html>. (Accessed: 9 December 2010).

Zhu, H. (2005) *Software Design Methodology: From Principles To Architectural Styles*. Oxford, Boston: Elsevier Butterworth-Heinemann.

Zhu, Y. (2007) *Measuring Effective Data Visualisation*, pp. 652-661. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2007.

X-Ways Software Technology AG (2010) WinHex: Features and Ways of Application. Available at: <http://www.winhex.com/winhex/allfeatures.html>. (Accessed: 9 December 2010).

## APPENDIX A

# EVENT LOGS DATA STRUCTURE FOR WINDOWS VISTA

Table A.1: Event Logs Header

No.	Offset (bytes )	Type	Description
1	0x00	char[8]	Magic string ("ElfFile") 0x00
2	0x08	char[8]	Unknown, const 0x00
3	0x10	int64	Number of current chunk
3	0x18	int64	Number of next record
4	0x20	uint32	Header space used, constant 0x80
5	0x24	uint16	Minor version, constant 1
6	0x26	uint16	Major version, constant 3
7	0x28	uint16	Size of header, constant 0x1000
8	0x2a	uint16	Chunk count
9	0x2c	char[76]	Unknown, const 0x00
10	0x78	uint32	Flags
11	0x7c	uint32	Check sum

Table A.2: Event Logs Chunk Header

No.	Offset (bytes )	Size (bytes)	Description
1	0x000	char[8]	Magic string ("ElfChnk") 0x00
2	0x008	int64	Number of first record in log
3	0x010	int64	Number of last record in log
4	0x018	int64	Number of first record in file
5	0x020	int64	Number of last record in file
6	0x028	uint32	Size of header
7	0x02c	uint32	Offset of last record
8	0x030	uint32	Offset of next record
9	0x034	uint32	DataCRC
10	0x038	char[68]	Unknown
11	0x07c	uint32	HeaderCRC
12	0x080	uint32[64]	StringTable
13	0x180	uint32[32]	TemplateTable

Table A.3: Event Logs Record

No.	Offset (bytes )	Type	Description
1	0x00	char[4]	Magic, constant 0x42, 0x00
2	0x04	uint32	Record length
3	0x08	int64	NumLogRecord
4	0x10	FILETIME	TimeCreated (FILETIME)
5	var.	char[]	Event message, binary XML stream
6	var.	uint32	Length (repeated)

## APPENDIX B

### REGISTRY DATA STRUCTURE

Table B.1: Security Records (SK) Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	2	String("sk")	Magic number
0x4	4	Offset	Pointer to previous SK record
0x8	4	Offset	Pointer to next SK record
0xC	4	Unsigned Integer	Reference count
0x10	4	Unsigned Integer	Size of security descriptor
0x14	Varies	Windows security descriptor	Data structure which contains owner SID, DACL, SACL and control flags.

Table B.2: Key Records (NK) Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	2	String("nk")	Magic number
0x2	2	Flags	0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080, 0x1000, 0x4000
0x4	8	Unsigned Integer	64-bit NT time stamp
0x10	4	Offset	Parent NK record
0x14	4	Unsigned Integer	Number of subkeys (stable)
0x18	4	Unsigned Integer	Number of subkeys (volatile)
0x1C	4	Offset	Pointer to subkey-list (stable)
0x20	4	Offset	Pointer to subkey-list (volatile)
0x24	4	Unsigned Integer	Number of values
0x28	4	Offset	Pointer to value-list for values
0x2C	4	Offset	Pointer to the SK record
0x30	4	Offset	Pointer to the class name
0x34	4	Unsigned Integer	Maximum number of bytes in a subkey name



Table B.2: Key Records (NK) Data Structures (Morgan, 2009) (continued)

Offset	Size (bytes)	Type	Description
0x38	4	Unsigned Integer	Maximum subkey class name length
0x3C	4	Unsigned Integer	Maximum number of bytes in a value name
0x40	4	Unsigned Integer	Maximum value data size
0x48	2	Unsigned Integer	Key name length
0x4A	2	Unsigned Integer	Class name length
0x4C	Variable	String	The key name; stored in ASCII and is typically NULL terminated

Table B.3: Subkey-lists Record Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	2	String	Magic number ("lf", "lh", "ri" or "li")
0x2	2	Unsigned Integer	Number of elements in this subkey-list
0x4	4 or 8 (each)	Structure List	Multiple subkey-list elements

Table B.4: Value Records (VK) Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	2	String("vk")	Magic number
0x2	2	Unsigned Integer	Value name length
0x4	4	Unsigned Integer	Data length
0x8	4	Offset	Pointer to data

Table B.4: Value Records (VK) Data Structures (Morgan, 2009) (continued)

Offset	Size (bytes)	Type	Description
0xC	4	Enumeration	Value type; One of; REG_NONE (0), REG_SZ (1), REG_EXPAND_SZ (2), REG_BINARY (3), REG_DWORD (4), REG_DWORD_BIG_ENDIAN (5), REG_LINK (6), REG_MULTI_SZ (7), REG_RESOURCE_LIST (8), REG_FULL_RESOURCE_DESCRIPTOR (9), REG_RESOURCE_REQUIREMENTS_LIST (10), REG_QWORD (11).
0x10	2	Flags	If the 0 bit is set, the value name is in ASCII, otherwise it is in UTF-16LE
0x14	Variable	String	The value name; stored in ASCII and is typically NULL terminated

Table B.5: Value-lists Records Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0..[4*(num. values)]	4	Offset	List of pointers to VK records; appear in order of value creation

Table B.6: Normal Data Blocks Records Data Structures Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	Variable	Raw Data	Data type and structure depends on type indicated by VK record

Table B.7: Big Data Records Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0	2	String ("db")	Magic number
0x2	2	Unsigned Integer	Number of data fragment
0x4	4	Offset	Pointer to big data indirect cell

Table B.8: Big Data Indirect Cells Records Data Structures (Morgan, 2009)

Offset	Size (bytes)	Type	Description
0x0..[4*(num. fragments)]	4	Offset	To a data fragment

# APPENDIX C

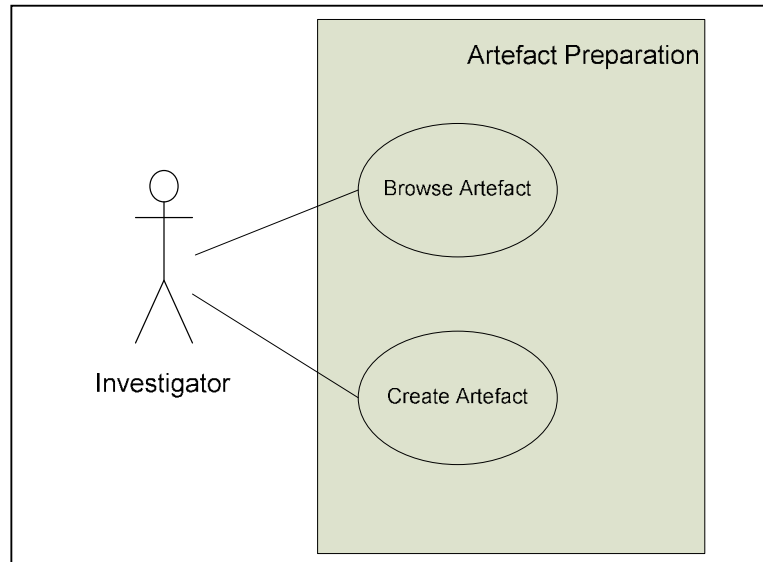
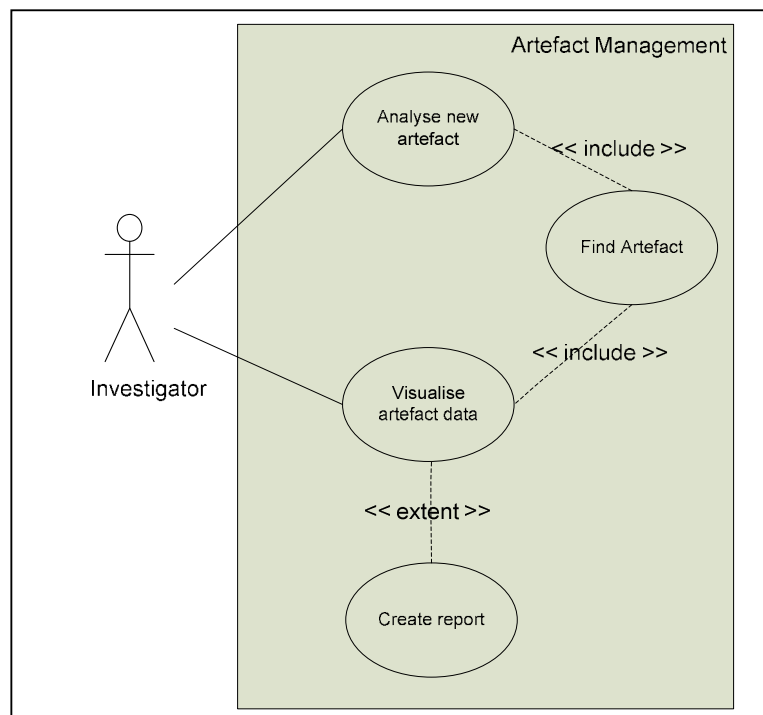
## USE CASES

Requirements List

Requirement	Use Case
To locate and access the artefact file in the file system.	Analyse new artefact
To create new artefact object.	
To extract data from the artefact's internal structure (extract fields).	
To record details of each artefact for each user. This will include the title of the artefact, the interpretation and reconstruction of the fields.	
To provide data store for insertion of data for each artefact.	
To get a database specification from the user, and tell the database object to read in an artefact object from the database.	Visualise artefact data
To display the current information about the desired artefact.	
To provide information that is used to visualise the data of an artefact in an intuitive format, that is to check the artefact data.	

Use Case Description

Use Case	Description
Analyse new artefact	<p>When a user gets the system artefact analysis task, details of the new artefact are entered into the database. These include the translation of a stream of bytes into a usable structure to recover evidence (data structures) from the artefact. The aim is to extract all data from any complex structure. Availability of a data store ensures retrieval and insertion of data.</p> <p>The investigator for that artefact is the person who analyses it.</p>
Visualise artefact data	<p>The artefact data may be examined to identify valuable meaningful data and allow for further analysis. The mapping of data to a visual form presents the data in a more human friendly format by the use of graphs, charts or illustrations.</p>

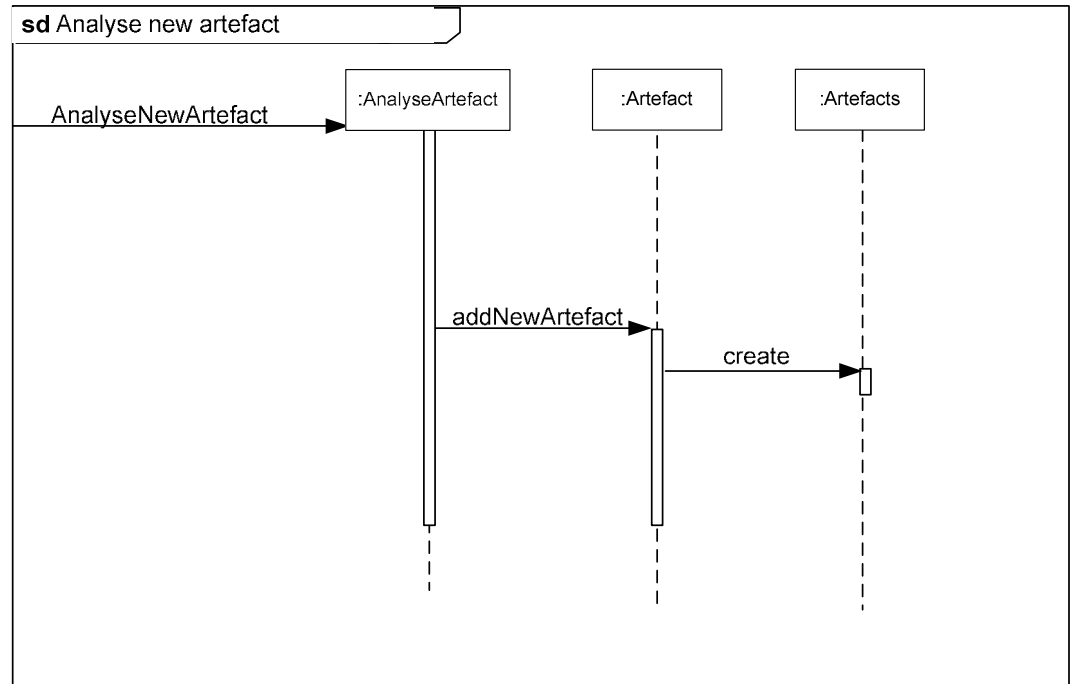
Artefact Preparation Use CasesArtefact Management Use Cases

## APPENDIX D

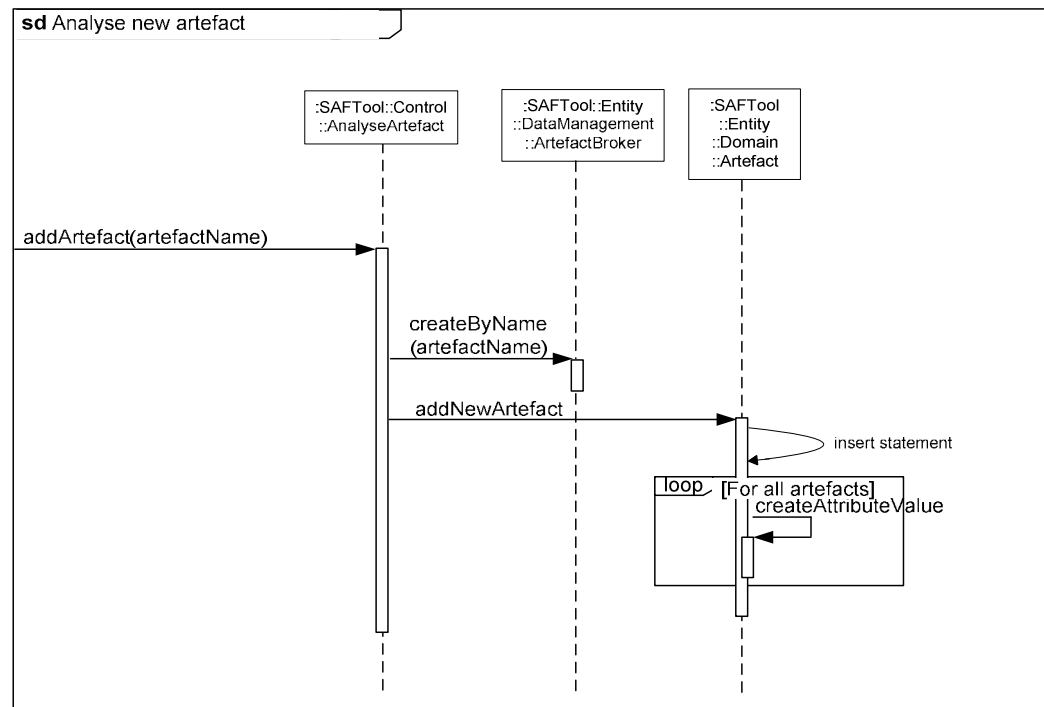
### SEQUENCE DIAGRAMS

Sequence diagram for the use case Analyse new artefact  
(to illustrate the interaction between instances of classes)

Detailed interaction for use case Analyse new artefact



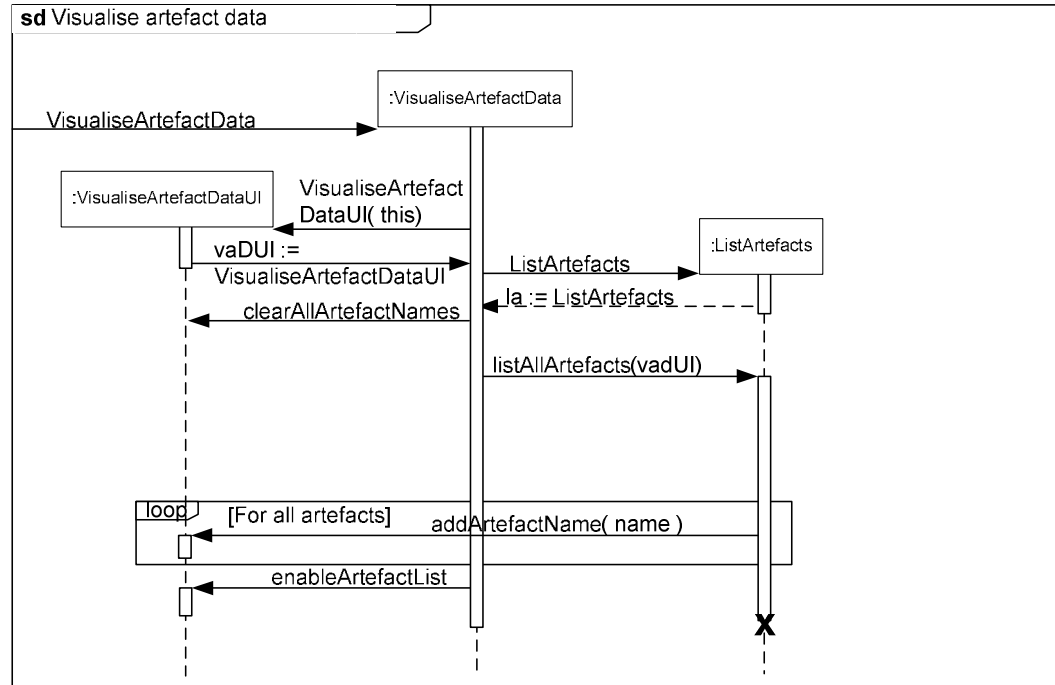
Sequence diagram for the operation **addArtefact ( )**



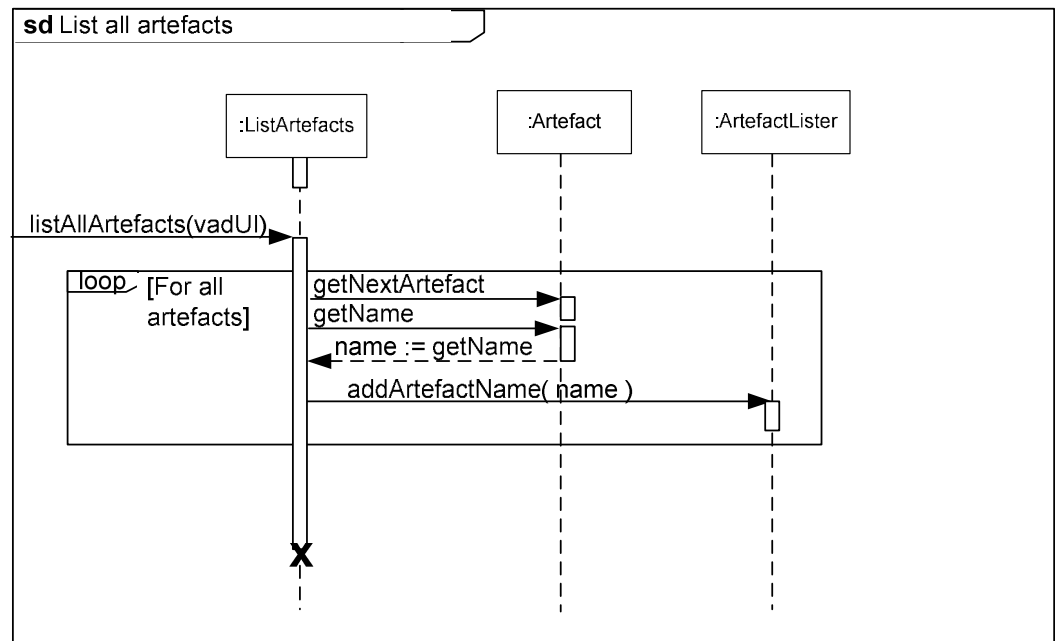


Sequence diagram for the use case Visualise artefact data  
(to illustrate the interaction between instances of classes)

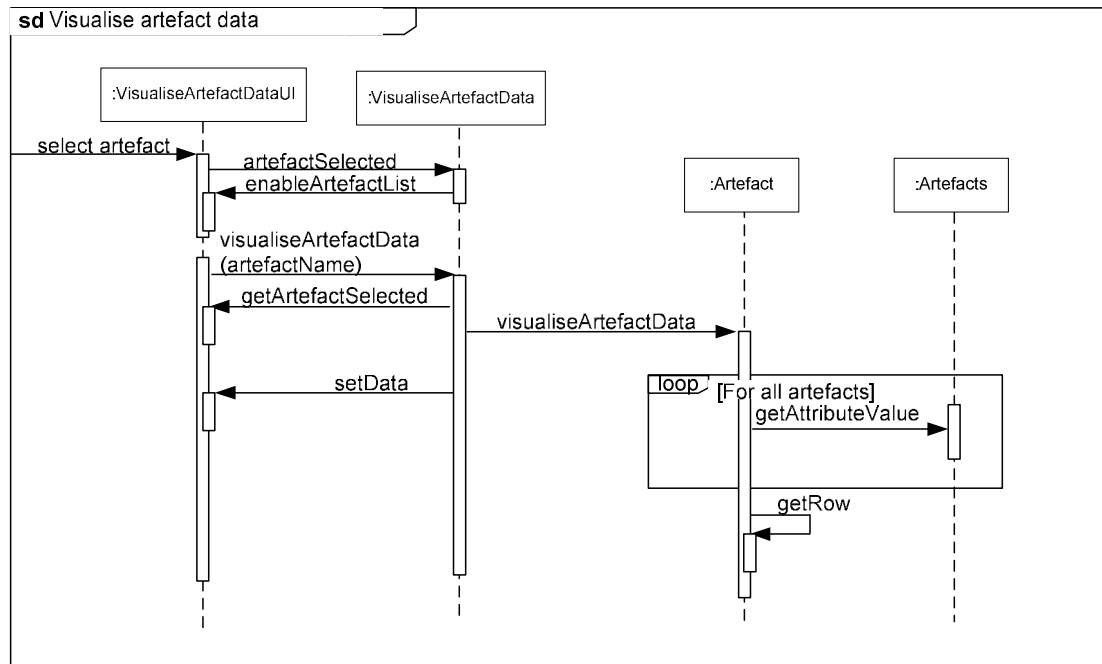
First part of detailed interaction for use case Visualise artefact data



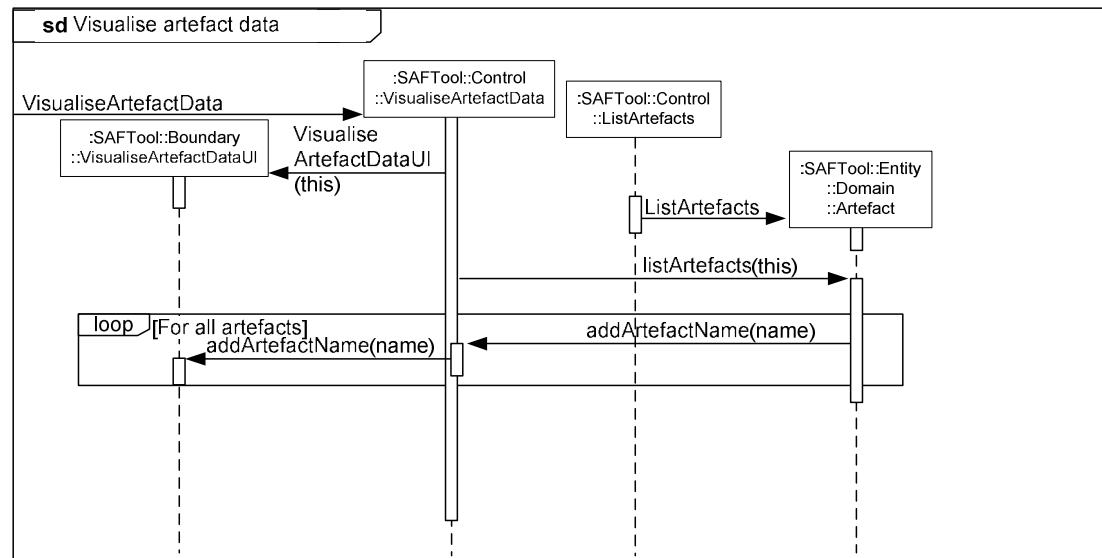
Sequence diagram for the operation `listAllArtefacts ( )`



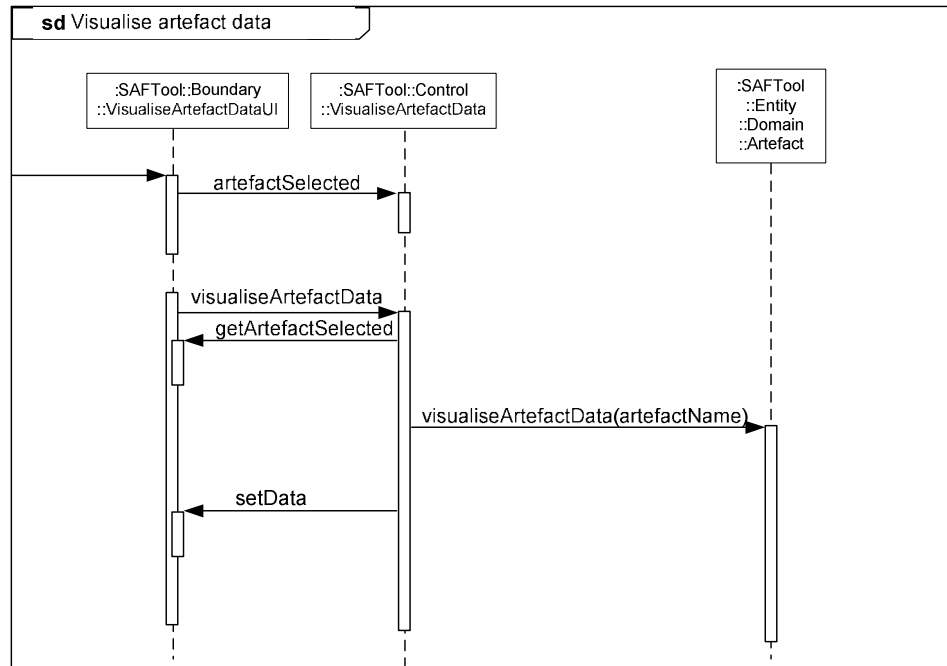
Final part of detailed interaction for use case Visualise artefact data



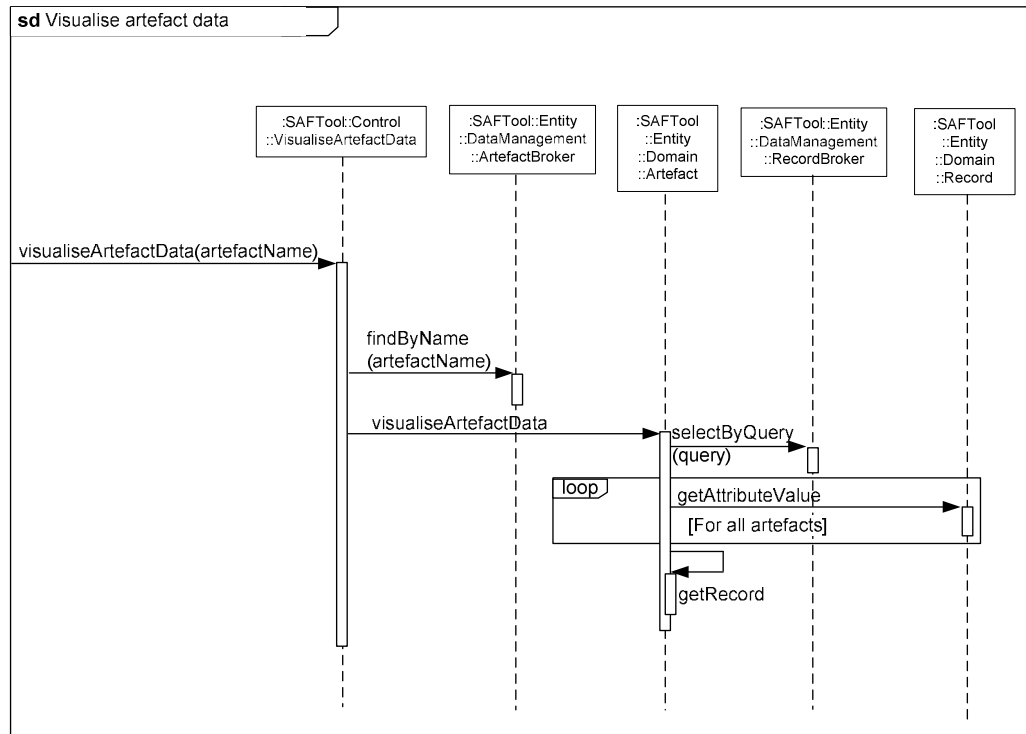
First part of detailed interaction for use case Visualise artefact data



Final part of detailed interaction for use case Visualise artefact data



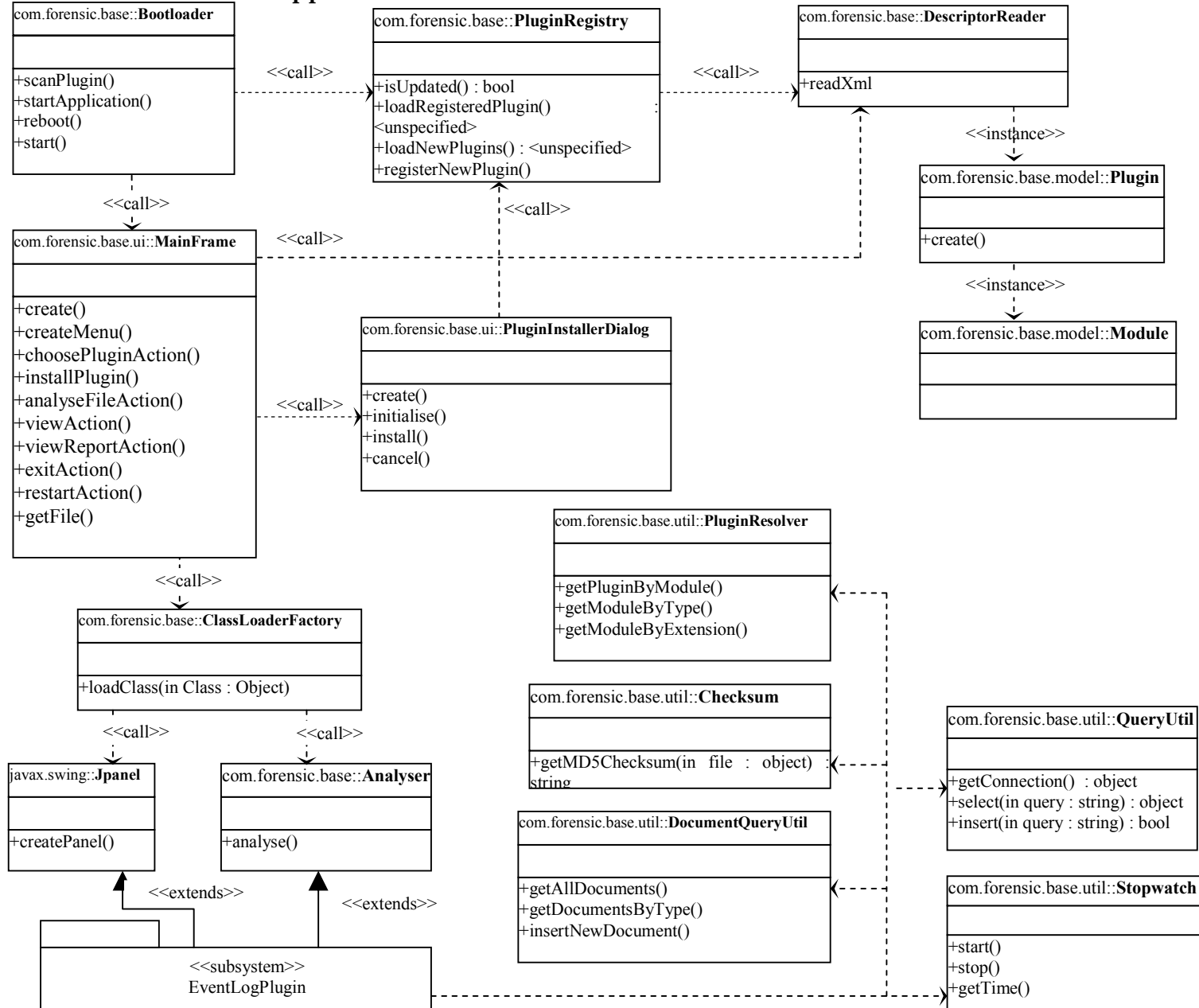
Sequence diagram for the operation `visualiseArtefactData ( )`



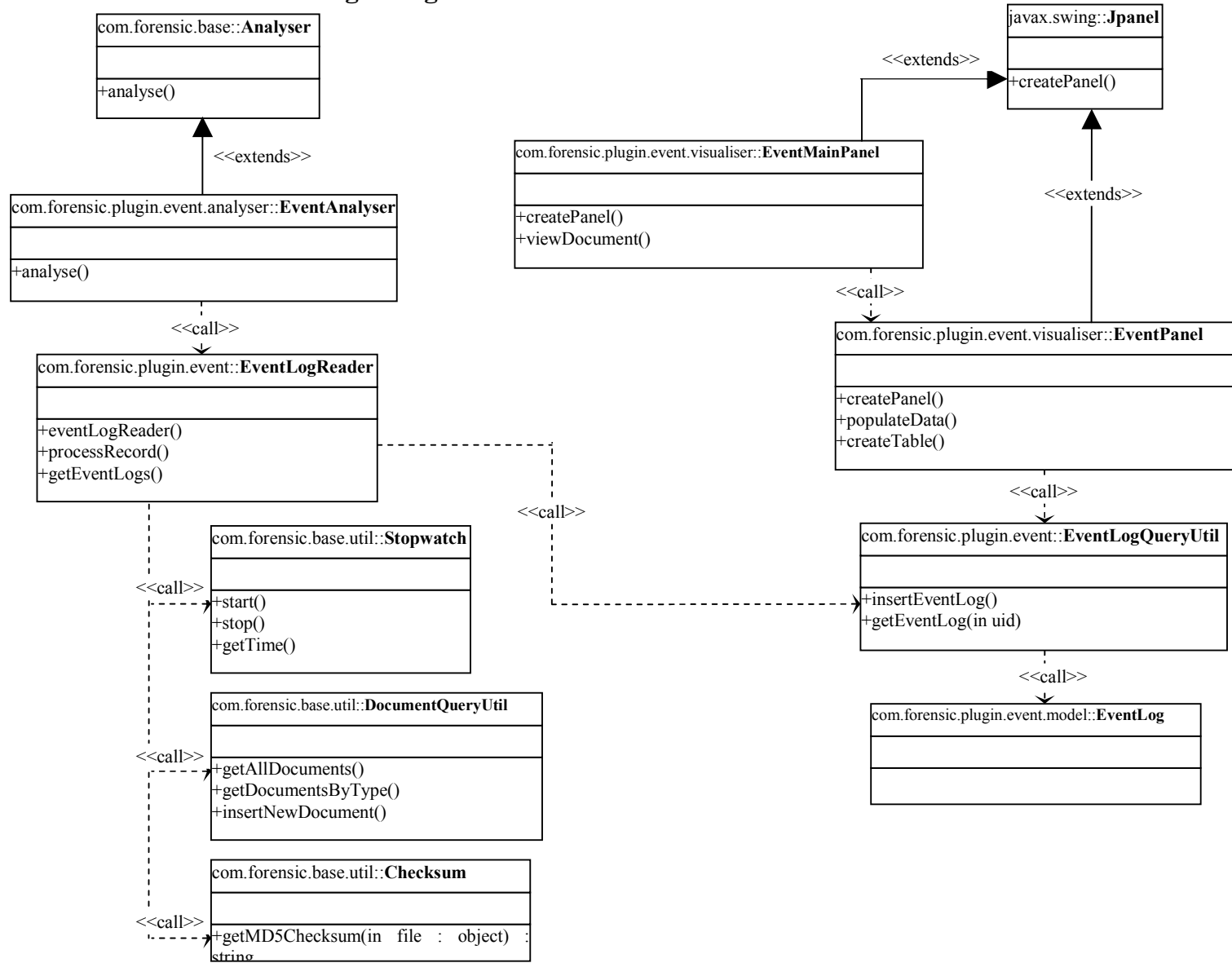
## APPENDIX E

### FULL CLASS DIAGRAMS

### Classes for Base Application



## Classes for Event Logs Plugin



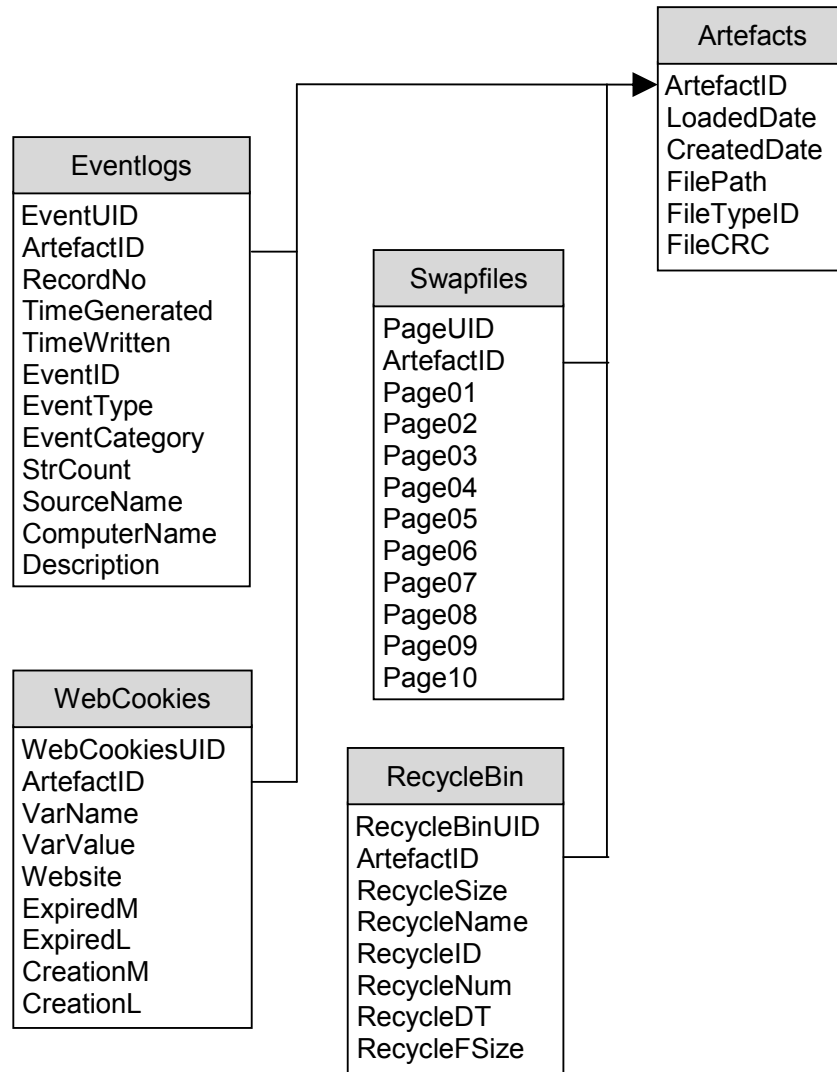
## Appendix E

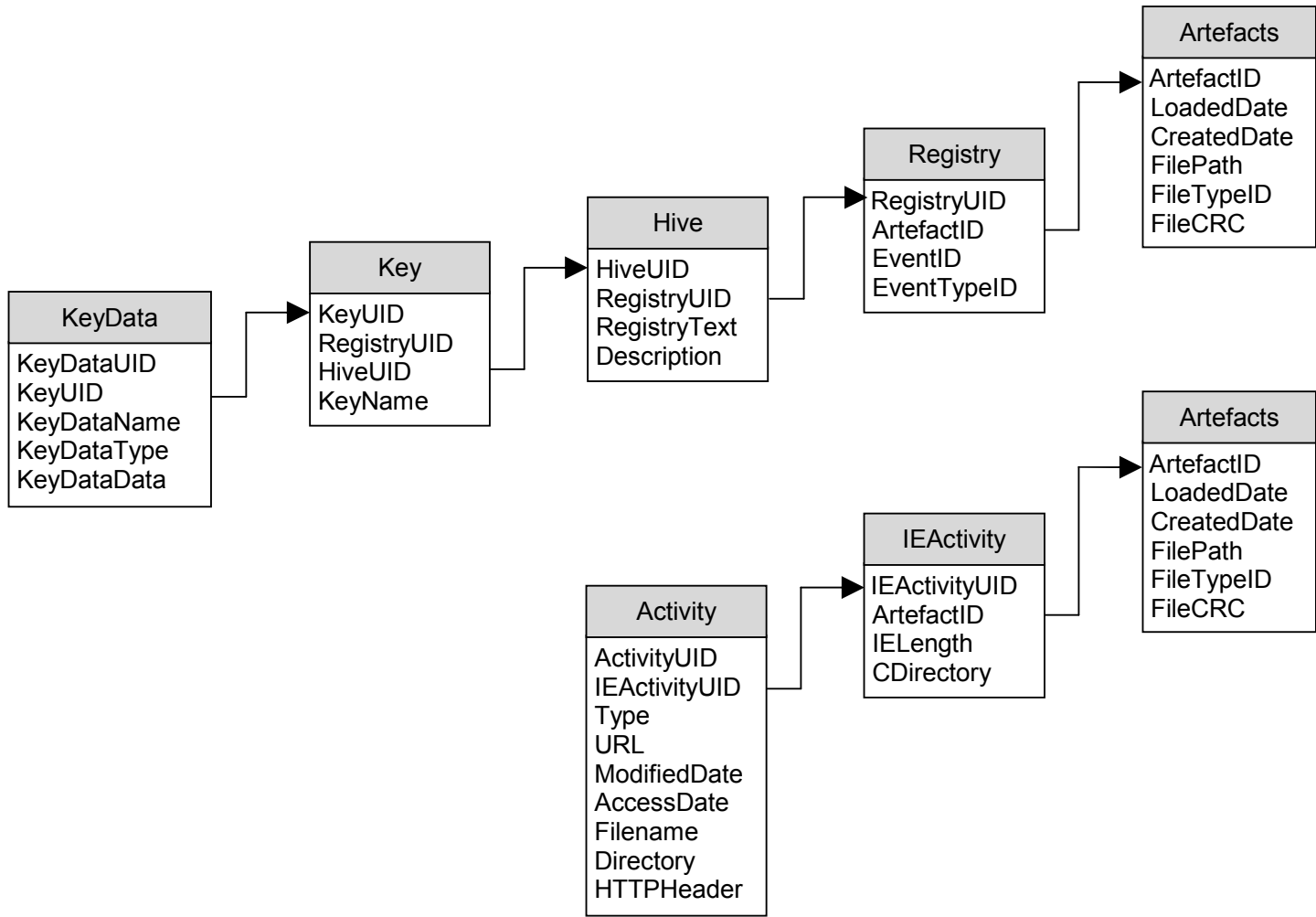


## APPENDIX F

### FULL DATABASE TABLES







```
CREATE TABLE Artefacts
(
    ArtefactID int NOT NULL PRIMARY KEY,
    LoadedDate datetime NOT NULL,
    CreatedDate datetime NOT NULL,
    FilePath varchar(200),
    FileType int NOT NULL)
FileCRC varchar(200)
);

CREATE TABLE Eventlog
(
    EventUID int NOT NULL PRIMARY KEY,
    ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
    RecordNo int NOT NULL,
    GeneratedTime datetime NOT NULL,
    WrittenTime datetime NOT NULL,
    EventID int NOT NULL,
    EventType int NOT NULL,
    EventCategory int NOT NULL,
    StrCount int NOT NULL,
    SourceName varchar(200) NULL,
    ComputerName varchar(200) NULL,
    Description varchar(200) NULL
);

CREATE TABLE Pagefile
(
    PageUID int NOT NULL PRIMARY KEY,
    ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
    Content blob NOT NULL,
    AsciiSize int NOT NULL,
    ChunkSize int NOT NULL,
    Block int NOT NULL
);
```

```
CREATE TABLE Registry
(
    RegistryUID int NOT NULL PRIMARY KEY,
    ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
    EventID int,
    EventTypeID int
);

CREATE TABLE Hive
(
    HiveUID int NOT NULL PRIMARY KEY,
    RegistryUID int NOT NULL FOREIGN KEY REFERENCES Registry(RegistryUID),
    Registry text NOT NULL,
    Description text NOT NULL
);

CREATE TABLE Key
(
    KeyUID int NOT NULL PRIMARY KEY,
    RegistryUID int NOT NULL FOREIGN KEY REFERENCES Registry(RegistryUID),
    HiveUID int NOT NULL FOREIGN KEY REFERENCES Hive(HiveUID),
    KeyName text NOT NULL
);

CREATE TABLE KeyData
(
    KeyDataUID int NOT NULL PRIMARY KEY,
    KeyUID int NOT NULL FOREIGN KEY REFERENCES Key(KeyUID),
    KeyDataName text NOT NULL,
    KeyDataType text,
    KeyDataData text
);
```

```
CREATE TABLE WebCookies
```

```
(
  WebCookiesUID int NOT NULL PRIMARY KEY,
  ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
  VarName varchar,
  VarValue varchar,
  Website varchar,
  ExpiredM timestamp,
  ExpiredL timestamp,
  CreationM timestamp,
  CreationL timestamp
);
```

```
CREATE TABLE RecycleBin
```

```
(
  RecycleBinUID int NOT NULL PRIMARY KEY,
  ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
  RecycleSize int,
  RecycleName varchar,
  RecycleID int,
  RecycleNum int,
  RecycleDT timestamp,
  RecycleFSize int
);
```

```
CREATE TABLE IEActivity
```

```
(
  IEActivityUID int NOT NULL PRIMARY KEY,
  ArtefactID int NOT NULL FOREIGN KEY REFERENCES Artefacts(ArtefactID),
  IELength int,
  CDirectory varchar
);
```

```
CREATE TABLE Activity
```

```
(
  ActivityUID int NOT NULL PRIMARY KEY,
  IEActivityUID int NOT NULL FOREIGN KEY REFERENCES IEActivity(IEActivityUID),
  Type varchar,
  URL varchar,
  ModifiedDate timestamp,
  AccessDate timestamp,
  Filename varchar,
  Directory varchar,
  HTTPHeader varchar
);
```

# APPENDIX G

## FULL DATASET

Table G.1: Event Logs File Name and File Size

No.	Event logs File Name	File Size (KB)
1.	AppEvent01.Evt	512
2.	SecEvent01.Evt	64
3.	SysEvent01.Evt	512
4.	AppEvent02.Evt	512
5.	SecEvent02.Evt	512
6.	SysEvent02.Evt	152
7.	AppEvent03.Evt	64
8.	SecEvent03.Evt	64
9.	SysEvent03.Evt	192
10.	AppEvent04.Evt	64
11.	SecEvent04.Evt	512
12.	SysEvent04.Evt	64

Table G.2: Swap Files File Name and File Size

No.	Swap Files File Name	File Size (KB)
1.	pagefile01.sys	1,506,576
2.	pagefile02.sys	540,672
3.	pagefile03.sys	786,432
4.	pagefile04.sys	774,144
5.	pagefile05.sys	117,760
6.	pagefile06.sys	512,000
7.	pagefile07.sys	173,648
8.	pagefile08.sys	196,608
9.	pagefile09.sys	393,216
10.	pagefile10.sys	393,216

A review of the tasks carried out by five participants when they were given four data files from the dataset to analyse and visualise.

Table G.3: Review of the Tasks Carried Out by Five Participants

No.	The Tasks	Data from the Dataset	Total number of tasks carried out by five people (95 tasks)
1	The built in extensible architecture of the SAFTool (6 tasks)		
	Click File from menu	no data is used	6 tasks X 5 = 30 tasks
	Click Analyse from Menu	no data is used	
	Click Visualise from Menu	no data is used	
	Click Report from Menu	no data is used	
	Click Install Plugin – EventLogPlugin.xml	no data is used	
	Click Install Plugin – PageFilePlugin.xml	no data is used	
2	Event Logs – 3 tasks for each data (3 tasks X 3 = 9 tasks)		
	Click Analyse Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	9 tasks X 5 = 45 tasks
	Click Visualise Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	
	Click Report Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	
3	Pagefile – 4 tasks for each data (4 tasks X 1 = 4 tasks)		
	Click Analyse Pagefile	pagefile10.sys	4 tasks X 5 = 20 tasks
	Click Visualise Pagefile	pagefile10.sys	
	Click Visualise Pagefile > Search keyword	pagefile10.sys	
	Click Report Pagefile	pagefile10.sys	



# APPENDIX H

## ADDITIONAL DATASET

Table H.1: Event Logs File Name and File Size of File Subjects in Consistency Test

No.	Event logs File Name	File Size (KB)
1.	AppEvent05.Evt	64
2.	SecEvent05.Evt	64
3.	SysEvent05.Evt	128
4.	AppEvent06.Evt	64
5.	SecEvent06.Evt	64
6.	SysEvent06.Evt	64
7.	AppEvent07.Evt	64
8.	SecEvent07.Evt	64
9.	SysEvent07.Evt	256
10.	AppEvent08.Evt	192
11.	SecEvent08.Evt	64
12.	SysEvent08.Evt	512
13.	AppEvent09.Evt	192
14.	SecEvent09.Evt	512
15.	SysEvent09.Evt	128
16.	AppEvent10.Evt	64
17.	SecEvent10.Evt	64
18.	SysEvent10.Evt	64
19.	AppEvent11.Evt	64
20.	SecEvent11.Evt	64
21.	SysEvent11.Evt	128
22.	AppEvent12.Evt	64
23.	SecEvent12.Evt	64
24.	SysEvent12.Evt	64
25.	AppEvent13.Evt	64
26.	SecEvent13.Evt	64
27.	SysEvent13.Evt	64
28.	AppEvent14.Evt	64
29.	SecEvent14.Evt	64
30.	SysEvent14.Evt	64

Table H.2: Swap Files File Name and File Size of File Subjects in Consistency Test

No.	Swap Files File Name	File Size (KB)
1.	pagefile11.sys	196,608
2.	pagefile12.sys	589,824
3.	pagefile13.sys	1,560,576
4.	pagefile14.sys	589,824
5.	pagefile15.sys	393,216
6.	pagefile16.sys	540,672
7.	pagefile17.sys	117,760
8.	pagefile18.sys	512,000
9.	pagefile19.sys	774,144
10.	pagefile20.sys	786,432

Table H.3: Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTTool (First Run)

Test Data File Name	Size (KB)	No. of Records		
		Event Viewer	evtstats.pl	SAFTTool
AppEvent05.Evt	64	X	112	112
SecEvent05.Evt	64	X	58	58
SysEvent05.Evt	128	X	347	347
AppEvent06.Evt	64	X	76	76
SecEvent06.Evt	64	X	0	0
SysEvent06.Evt	64	X	262	262
AppEvent07.Evt	64	X	123	123
SecEvent07.Evt	64	X	260	260
SysEvent07.Evt	256	X	370	370
AppEvent08.Evt	192	X	139	139
SecEvent08.Evt	64	X	0	0
SysEvent08.Evt	512	X	1547	1547
AppEvent09.Evt	192	X	106	106
SecEvent09.Evt	512	X	2317	2317
SysEvent09.Evt	128	X	395	395
AppEvent10.Evt	64	X	63	63
SecEvent10.Evt	64	X	0	0
SysEvent10.Evt	64	X	295	295
AppEvent11.Evt	64	X	187	187
SecEvent11.Evt	64	X	0	0
SysEvent11.Evt	128	X	359	359
AppEvent12.Evt	64	X	110	110
SecEvent12.Evt	64	X	206	206
SysEvent12.Evt	64	X	236	236
AppEvent13.Evt	64	X	89	89
SecEvent13.Evt	64	X	0	0
SysEvent13.Evt	64	X	247	247
AppEvent14.Evt	64	X	76	76
SecEvent14.Evt	64	X	124	124
SysEvent14.Evt	64	X	312	312
X – unable to complete the operation. The event log file is reported as corrupted.				

Table H.4: Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTTool (Second Run)

Test Data File Name	Size (KB)	No. of Records		
		Event Viewer	evtstats.pl	SAFTTool
AppEvent05.Evt	64	X	112	112
SecEvent05.Evt	64	X	58	58
SysEvent05.Evt	128	X	347	347
AppEvent06.Evt	64	X	76	76
SecEvent06.Evt	64	X	0	0
SysEvent06.Evt	64	X	262	262
AppEvent07.Evt	64	X	123	123
SecEvent07.Evt	64	X	260	260
SysEvent07.Evt	256	X	370	370
AppEvent08.Evt	192	X	139	139
SecEvent08.Evt	64	X	0	0
SysEvent08.Evt	512	X	1547	1547
AppEvent09.Evt	192	X	106	106
SecEvent09.Evt	512	X	2317	2317
SysEvent09.Evt	128	X	395	395
AppEvent10.Evt	64	X	63	63
SecEvent10.Evt	64	X	0	0
SysEvent10.Evt	64	X	295	295
AppEvent11.Evt	64	X	187	187
SecEvent11.Evt	64	X	0	0
SysEvent11.Evt	128	X	359	359
AppEvent12.Evt	64	X	110	110
SecEvent12.Evt	64	X	206	206
SysEvent12.Evt	64	X	236	236
AppEvent13.Evt	64	X	89	89
SecEvent13.Evt	64	X	0	0
SysEvent13.Evt	64	X	247	247
AppEvent14.Evt	64	X	76	76
SecEvent14.Evt	64	X	124	124
SysEvent14.Evt	64	X	312	312
X – unable to complete the operation. The event log file is reported as corrupted.				

Table H.5: Number of Records Retrieved for Event Viewer, evtstats.pl and SAFTool (Third Run)

Test Data File Name	Size (KB)	No. of Records		
		Event Viewer	evtstats.pl	SAFTool
AppEvent05.Evt	64	X	112	112
SecEvent05.Evt	64	X	58	58
SysEvent05.Evt	128	X	347	347
AppEvent06.Evt	64	X	76	76
SecEvent06.Evt	64	X	0	0
SysEvent06.Evt	64	X	262	262
AppEvent07.Evt	64	X	123	123
SecEvent07.Evt	64	X	260	260
SysEvent07.Evt	256	X	370	370
AppEvent08.Evt	192	X	139	139
SecEvent08.Evt	64	X	0	0
SysEvent08.Evt	512	X	1547	1547
AppEvent09.Evt	192	X	106	106
SecEvent09.Evt	512	X	2317	2317
SysEvent09.Evt	128	X	395	395
AppEvent10.Evt	64	X	63	63
SecEvent10.Evt	64	X	0	0
SysEvent10.Evt	64	X	295	295
AppEvent11.Evt	64	X	187	187
SecEvent11.Evt	64	X	0	0
SysEvent11.Evt	128	X	359	359
AppEvent12.Evt	64	X	110	110
SecEvent12.Evt	64	X	206	206
SysEvent12.Evt	64	X	236	236
AppEvent13.Evt	64	X	89	89
SecEvent13.Evt	64	X	0	0
SysEvent13.Evt	64	X	247	247
AppEvent14.Evt	64	X	76	76
SecEvent14.Evt	64	X	124	124
SysEvent14.Evt	64	X	312	312
X – unable to complete the operation. The event log file is reported as corrupted.				

Table H.6: Processing Time for Event Viewer, lsevt.pl and SAFTool  
(First Run)

Test Data File Name	Size (KB)	Event Viewer Processing Time (ms)	lsevt.pl		SAFTool	
			No. of Records	Processing Time (ms)	No. of Records	Processing Time (ms)
AppEvent05.Evt	64	X	112	76	112	703
SecEvent05.Evt	64	X	58	53	58	359
SysEvent05.Evt	128	X	347	182	347	1328
AppEvent06.Evt	64	X	76	61	76	422
SecEvent06.Evt	64	X	0	0	0	0
SysEvent06.Evt	64	X	262	150	262	1281
AppEvent07.Evt	64	X	123	87	123	719
SecEvent07.Evt	64	X	260	148	260	1063
SysEvent07.Evt	256	X	370	386	370	1468
AppEvent08.Evt	192	X	139	96	139	812
SecEvent08.Evt	64	X	0	0	0	0
SysEvent08.Evt	512	X	1547	807	1547	5906
AppEvent09.Evt	192	X	106	76	106	453
SecEvent09.Evt	512	X	2317	839	2317	10797
SysEvent09.Evt	128	X	395	393	395	2805.66
AppEvent10.Evt	64	X	63	56	63	349
SecEvent10.Evt	64	X	0	0	0	0
SysEvent10.Evt	64	X	295	145	295	1251
AppEvent11.Evt	64	X	187	100	187	843
SecEvent11.Evt	64	X	0	0	0	0
SysEvent11.Evt	128	X	359	284	359	1583.66
AppEvent12.Evt	64	X	110	64	110	547
SecEvent12.Evt	64	X	206	120	206	901
SysEvent12.Evt	64	X	236	140	236	844
AppEvent13.Evt	64	X	89	76	89	468.66
SecEvent13.Evt	64	X	0	0	0	0
SysEvent13.Evt	64	X	247	148	247	1067.66
AppEvent14.Evt	64	X	76	70	76	422
SecEvent14.Evt	64	X	124	95	124	828
SysEvent14.Evt	64	X	312	154	312	1385.33

X – unable to complete the operation. The event log file is reported as corrupted.

Table H.7: Processing Time for Event Viewer, lsevt.pl and SAFTool  
(Second Run)

Test Data File Name	Size (KB)	Event Viewer Processing Time (ms)	lsevt.pl		SAFTool	
			No. of Records	Processing Time (ms)	No. of Records	Processing Time (ms)
AppEvent05.Evt	64	X	112	98	112	625
SecEvent05.Evt	64	X	58	62	58	328
SysEvent05.Evt	128	X	347	229	347	1344
AppEvent06.Evt	64	X	76	62	76	453
SecEvent06.Evt	64	X	0	0	0	0
SysEvent06.Evt	64	X	262	148	262	1187
AppEvent07.Evt	64	X	123	87	123	625
SecEvent07.Evt	64	X	260	150	260	1094
SysEvent07.Evt	256	X	370	406	370	1735
AppEvent08.Evt	192	X	139	109	139	875
SecEvent08.Evt	64	X	0	0	0	0
SysEvent08.Evt	512	X	1547	807	1547	6031
AppEvent09.Evt	192	X	106	78	106	453
SecEvent09.Evt	512	X	2317	864	2317	8969
SysEvent09.Evt	128	X	395	410	395	1296
AppEvent10.Evt	64	X	63	54	63	266
SecEvent10.Evt	64	X	0	0	0	0
SysEvent10.Evt	64	X	295	159	295	987
AppEvent11.Evt	64	X	187	129	187	906
SecEvent11.Evt	64	X	0	0	0	0
SysEvent11.Evt	128	X	359	214	359	1688
AppEvent12.Evt	64	X	110	68	110	516
SecEvent12.Evt	64	X	206	120	206	844
SysEvent12.Evt	64	X	236	143	236	922
AppEvent13.Evt	64	X	89	76	89	453
SecEvent13.Evt	64	X	0	0	0	0
SysEvent13.Evt	64	X	247	145	247	1109
AppEvent14.Evt	64	X	76	70	76	360
SecEvent14.Evt	64	X	124	100	124	781
SysEvent14.Evt	64	X	312	173	312	1516

X – unable to complete the operation. The event log file is reported as corrupted.

Table H.8: Processing Time for Event Viewer, lsevt.pl and SAFTool  
(Third Run)

Test Data File Name	Size (KB)	Event Viewer Processing Time (ms)	lsevt.pl		SAFTTool	
			No. of Records	Processing Time (ms)	No. of Records	Processing Time (ms)
AppEvent05.Evt	64	X	112	87	112	656
SecEvent05.Evt	64	X	58	56	58	344
SysEvent05.Evt	128	X	347	185	347	1500
AppEvent06.Evt	64	X	76	78	76	437
SecEvent06.Evt	64	X	0	0	0	0
SysEvent06.Evt	64	X	262	150	262	1250
AppEvent07.Evt	64	X	123	89	123	687
SecEvent07.Evt	64	X	260	146	260	1110
SysEvent07.Evt	256	X	370	398	370	1984
AppEvent08.Evt	192	X	139	109	139	828
SecEvent08.Evt	64	X	0	0	0	0
SysEvent08.Evt	512	X	1547	800	1547	6140
AppEvent09.Evt	192	X	106	82	106	516
SecEvent09.Evt	512	X	2317	850	2317	9500
SysEvent09.Evt	128	X	395	89	395	719
AppEvent10.Evt	64	X	63	70	63	265
SecEvent10.Evt	64	X	0	0	0	0
SysEvent10.Evt	64	X	295	146	295	1188
AppEvent11.Evt	64	X	187	118	187	875
SecEvent11.Evt	64	X	0	0	0	0
SysEvent11.Evt	128	X	359	292	359	719
AppEvent12.Evt	64	X	110	123	110	532
SecEvent12.Evt	64	X	206	115	206	1015
SysEvent12.Evt	64	X	236	126	236	969
AppEvent13.Evt	64	X	89	75	89	485
SecEvent13.Evt	64	X	0	0	0	0
SysEvent13.Evt	64	X	247	150	247	1016
AppEvent14.Evt	64	X	76	70	76	422
SecEvent14.Evt	64	X	124	96	124	953
SysEvent14.Evt	64	X	312	159	312	1375

X – unable to complete the operation. The event log file is reported as corrupted.



Table H.9: Total Size of Data for WinHex and the SAFTool (First Run)

Test Data File Name	Size (KB)	Total Size of Data (bytes)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	1,598,029,824	1,598,029,824
pagefile12.sys	589,824	603,979,776	603,979,776
pagefile13.sys	2,095,104	2,145,386,496	2,145,386,496
pagefile14.sys	786,432	805,306,368	805,306,368
pagefile15.sys	196,608	201,326,592	201,326,592
pagefile16.sys	540,672	553,648,128	553,648,128
pagefile17.sys	117,760	120,586,240	120,586,240
pagefile18.sys	512,000	524,288,000	524,288,000
pagefile19.sys	774,144	792,723,456	792,723,456
pagefile20.sys	393,216	402,653,184	402,653,184

Table H.10: Total Size of Data for WinHex and the SAFTool (Second Run)

Test Data File Name	Size (KB)	Total Size of Data (bytes)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	1,598,029,824	1,598,029,824
pagefile12.sys	589,824	603,979,776	603,979,776
pagefile13.sys	2,095,104	2,145,386,496	2,145,386,496
pagefile14.sys	786,432	805,306,368	805,306,368
pagefile15.sys	196,608	201,326,592	201,326,592
pagefile16.sys	540,672	553,648,128	553,648,128
pagefile17.sys	117,760	120,586,240	120,586,240
pagefile18.sys	512,000	524,288,000	524,288,000
pagefile19.sys	774,144	792,723,456	792,723,456
pagefile20.sys	393,216	402,653,184	402,653,184

Table H.11: Total Size of Data for WinHex and the SAFTool (Third Run)

Test Data File Name	Size (KB)	Total Size of Data (bytes)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	1,598,029,824	1,598,029,824
pagefile12.sys	589,824	603,979,776	603,979,776
pagefile13.sys	2,095,104	2,145,386,496	2,145,386,496
pagefile14.sys	786,432	805,306,368	805,306,368
pagefile15.sys	196,608	201,326,592	201,326,592
pagefile16.sys	540,672	553,648,128	553,648,128
pagefile17.sys	117,760	120,586,240	120,586,240
pagefile18.sys	512,000	524,288,000	524,288,000
pagefile19.sys	774,144	792,723,456	792,723,456
pagefile20.sys	393,216	402,653,184	402,653,184

Table H.12: Processing Time for WinHex and SAFTool (First Run)

Test Data File Name	Size (KB)	Processing Time (ms)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	37	439,890
pagefile12.sys	589,824	25	307,547
pagefile13.sys	2,095,104	34	572,719
pagefile14.sys	786,432	28	406,469
pagefile15.sys	196,608	25	239,969
pagefile16.sys	540,672	25	314,687
pagefile17.sys	117,760	20	240,797
pagefile18.sys	512,000	26	314,625
pagefile19.sys	774,144	32	340,297
pagefile20.sys	393,216	25	262,688

Table H.13: Processing Time for WinHex and SAFTool (Second Run)

Test Data File Name	Size (KB)	Processing Time (ms)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	18	638,844
pagefile12.sys	589,824	20	375,312
pagefile13.sys	2,095,104	21	680,156
pagefile14.sys	786,432	20	388,094
pagefile15.sys	196,608	18	318,469
pagefile16.sys	540,672	17	353,640
pagefile17.sys	117,760	18	295,860
pagefile18.sys	512,000	20	328,187
pagefile19.sys	774,144	18	436,360
pagefile20.sys	393,216	17	444,766

Table H.14: Processing Time for WinHex and SAFTool (Third Run)

Test Data File Name	Size (KB)	Processing Time (ms)	
		WinHex	SAFTool
pagefile11.sys	1,506,576	15	306,047
pagefile12.sys	589,824	18	216,578
pagefile13.sys	2,095,104	20	463,500
pagefile14.sys	786,432	20	285,125
pagefile15.sys	196,608	15	145,766
pagefile16.sys	540,672	20	219,406
pagefile17.sys	117,760	17	148,360
pagefile18.sys	512,000	15	227,343
pagefile19.sys	774,144	14	239,781
pagefile20.sys	393,216	17	142,562

Table H.15: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by WinHex (First Run)

Test Data File Name	Size (KB)	WinHex				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√

Table H.16: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by WinHex (Second Run)

Test Data File Name	Size (KB)	WinHex				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√

Table H.17: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by WinHex (Third Run)

Test Data File Name	Size (KB)	WinHex				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√

Table H.18: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by the SAFTool (First Run)

Test Data File Name	Size (KB)	SAFTool				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√

Table H.19: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by the SAFTool (Second Run)

Test Data File Name	Size (KB)	SAFTool				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√

Table H.20: Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ Terms in Swap Files Identified by the SAFTTool (Third Run)

Test Data File Name	Size (KB)	SAFTTool				
		Existence of ‘mail’, ‘from’, ‘www’, ‘html’ and ‘send’ term				
		mail	From	www	Html	send
pagefile11.sys	1,506,576	√	√	√	√	√
pagefile12.sys	589,824	√	√	√	√	√
pagefile13.sys	2,095,104	√	√	√	√	√
pagefile14.sys	786,432	√	√	√	√	√
pagefile15.sys	196,608	√	√	√	√	√
pagefile16.sys	540,672	√	√	√	√	√
pagefile17.sys	117,760	√	√	√	√	√
pagefile18.sys	512,000	√	√	√	√	√
pagefile19.sys	774,144	√	√	√	√	√
pagefile20.sys	393,216	√	√	√	√	√



A review of the tasks carried out by fifteen participants when they were given four data files from the dataset to analyse and visualise.

Table H.21: Review of the Tasks Carried Out by Fifteen Participants

No.	The Tasks	Data from the Dataset	Total number of tasks carried out by fifteen people (285 tasks)
1	The built in extensible architecture of the SAFTool (6 tasks)		
	Click File from menu	no data is used	6 tasks X 15 = 90 tasks
	Click Analyse from Menu	no data is used	
	Click Visualise from Menu	no data is used	
	Click Report from Menu	no data is used	
	Click Install Plugin – EventLogPlugin.xml	no data is used	
	Click Install Plugin – PageFilePlugin.xml	no data is used	
2	Event Logs – 3 tasks for each data (3 tasks X 3 = 9 tasks)		
	Click Analyse Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	9 tasks X 15 = 135 tasks
	Click Visualise Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	
	Click Report Event Logs	AppEvent04.Evt SecEvent04.Evt SysEvent04.Evt	
3	Pagefile – 4 tasks for each data (4 tasks X 1 = 4 tasks)		
	Click Analyse Pagefile	pagefile10.sys	4 tasks X 15 = 60 tasks
	Click Visualise Pagefile	pagefile10.sys	
	Click Visualise Pagefile > Search keyword	pagefile10.sys	
	Click Report Pagefile	pagefile10.sys	

# APPENDIX I

## QUESTIONNAIRE

## Questionnaires

### Questionnaires on user satisfaction level with the System Generated Artefacts Forensic Analysis Application.

Objectives of the questionnaires:

1. To gain an understanding and increase the level of awareness surrounding the use of System Generated Artefacts in an investigation.
2. To assess your level of satisfaction with the SAFTool's ability to aid your understanding of system generated artefacts and their data content.
3. To find out how easy you think it would be to modify and extend the tool.

Please score the performance measures using the following criteria:

**Very unsatisfied = 1; Unsatisfied = 2; Fair = 3; Satisfied = 4;**

**Very Satisfied = 5**

A. System Generated Artefacts are files which are created due to the routine operation of a computer operating system. These may be created without the user's knowledge.					
1. Data analysis and presentation of data in the application increases the understanding of the main points of the system generated artefacts and their data content. The use of the tool highlights the importance of system generated artefacts. Comment please.					
Comments					Please Tick
					1
					2
					3
					4
					5

2. The chosen analysis and display methods aid the understanding of the system generated artefacts and their data content. Comment please.					
Comments	Please Tick				
	1	2	3	4	5
3. Additional information annotation provided in the application aids the understanding of the information contained in the system generated artefacts. Comment please.					
Comments	Please Tick				
	1	2	3	4	5
<b>B. Information Representation</b>					
4. The data analysis and data visualisation easily and clearly indicate the content of the Windows system generated artefacts. Comment please.					
Comments	Please Tick				
	1	2	3	4	5

5. The data analysis and data visualisation used in the prototype help the user to understand the information contained in the Windows system generated artefacts. Comment please.					
Comments	Please Tick				
	1	2	3	4	5
C. Extensibility					
6. In your expert opinion, and given the information provided, how easy is it to modify and extend the tool?					
Comments	Please Tick				
	1	2	3	4	5

Name:

---

Age:

---

Gender:

---

Years of Experience in Digital Forensics Field:

---

List of Computer Forensics Tool Used:

---

Thank You