A Study of Application Level Information From The Volatile Memory of Windows Computer Systems

Funminiyi Akanfe Olajide

September 2011

This thesis is submitted in partial fulfilment of the requirement for the award of the degree of Doctor of Philosophy at the University of Portsmouth.

To Almighty God through Christ Jesus

and

To the memory of my late father, Elder S.O. Olajide and my late mother, Prophetess R.O. Olajide

Abstract

The purpose of this research work was to investigate into the seven most commonly used applications in order to uncover information that may have been hidden from forensic investigators by extracting the application level information from volatile memory of a Windows system and performing analysis of that volatile memory. The aim of this research was to formulate how the extracted application level information can be reconstructed to describe what user activities had taken place on the application under investigation. After reviewing the relevant literature on volatile memory analysis and forensically relevant data from Windows applications, this thesis confines its research to a study of the application level information and the volatile memory analysis of Windows applications.

Quantitative and qualitative results were produced in this study. The quantitative assessment consists of four metrics and that were used to investigate the quantity of user input on the applications while the qualitative measures were formulated to infer what the user is doing on the application, what they have been doing and what they are using the applications for. The reconstruction of user input activities was carried out by using some commonly used English words to search for user input and pattern matching techniques for when the user input is known in the investigation.

The analysis of user input was discussed based on four scenarios developed for this research. The result shows that different amounts of user input can be recovered from various applications. The result in scenario 1, indicates that user input can be recovered easily from Word, PowerPoint, Outlook Email and Internet Explorer 7.0 and that little user input can be found on Excel, MS Access and Adobe Reader 8.0. In scenario 2, a significant amount of user input was recovered in the memory allocated to all the applications except MS Access where little user input was found. In scenario3, only Outlook Email and Internet Explorer 7.0 resulted in a large amount of user input being recovered. The rest of the applications retain little user input in memory. In scenario 4, a greatly reduced amount of information was found for all the applications. But some user input was found from Outlook Email and Internet Explorer 7.0 which shows that user input can be retained for some time in the memory. After the analysis of user input, the importance of volatile memory of the application level information was discussed.

A procedure has been formulised for the extraction and analysis of application level information and these have been discussed with respect to their use in the court of law based on the five Daubert tests of scientific method of gathering digital evidence. As presented, three out of the Daubert tests have been completed while the two others forms the unique contribution of the research project to digital forensic community. The author recommends that the research theory of application level information should be extended to other operating systems using the scenarios formulated in this research project.

Acknowledgements

Firstly, I would like to thank the Almighty God through Jesus the Christ for helping me to be able to complete this PhD course. My appreciation and love goes to my wife and the children who have put up with me throughout the period of this PhD course. My wife heartfelt love, encouragement, support, prayers and care will remain indelible in my heart forever. I am especially grateful to God for my wife and my wonderful children for supporting me to pursue this degree and for providing a loving environment for me. Without God and my wife's encouragement, I would not have finished the degree.

I would like to express a deep thanks to my supervisor Dr. Nick Savage for his continued commitment to this project and his comments and suggestions. It has been an honour to be his first PhD student. He has taught me, both consciously and unconsciously, how good experimental research is done. I appreciate all his contributions of time and ideas to make my PhD experience productive and stimulating. I am also thankful for the excellent example he has provided as a successful Professor. I will forever be grateful to Dr. Nick Savage for supporting me and advising me to the conclusion of this PhD course. Also thanks to Dr. David Ndzi, Dr. David Sander, Dr. Misha Fillip, Dr. Shikun Zhou, Dr. Linda Yang and Professor Charles Shoniregun for their helpful suggestions. My gratitude goes to my research colleagues who have provided some help with computing problems. I will never forget the work atmosphere and the happy hours of social discussion and entertainment that I have enjoyed with my colleagues and friends at the upper room floor of the Anglesea Building at the School of Engineering.

Thanks must also go to Mr Tim and Mrs Alison McCann for their love, prayers, encouragement and their caring heart at the second and third year of this course. I would like to thank Fr Tom Grufferty for his kindness and prayers. Thanks also goes to David Willets, Jackie Scaddan, Mr Ian and Aunt Julia Corps for their love, care and prayers. I would also like to thank Mr Mathew Olawo, Mr Som Murkherjee and Mr Bashir Adetunji for their good wishes and prayers.

Finally, a special thanks to my brother Olabode Olajide, my God's brothers and sisters in the community and my friends for being patient.

List of Abbreviations

ALI – Application Level Information

ACPO - Association of Chief Police Officers

ASCII – American Standard Code for Information Interchange

CIBC - Canadian Imperial Bank of Commerce

CFIT - Computer Forensic Tool Testing

CPU – Central Processing Unit

DFRWS - Digital Forensic Research Workshops

DLL – Dynamic-Link Library

DF – Digital Forensic

DFSs - Digital Forensic Standards

DFTTI - Digital Forensics Tool Testing Images

DIPFALI - Digital Investigation Process Framework of Application Level Information

ECSI - Electronic Crime Scene Investigation

EWs - Expert Witnesses

FSR - Forensic Science Regulator

FRU - Forensic Regulator Unit

FSAC - Forensic Science Advisory Council

GE - Google Earth

GMs - Google Maps

GB – Gigabyte

HO - Home Office

ID – Identification

I/O – Input/Output

LSC - Legal Services Commission

MQ – MapQuest

MS – Microsoft

MB – Megabyte

NIST - National Institute of Standards and Technology

NHTCU - National Hi-Tech Crime Unit

NFRC - National Fraud Reporting Centre

NIJ - National Institute of Justice

OS – Operating System

PCB - Process Control Block

PID – Process Identification

PI - Process Identifier

PCEu – Processing Central E-Crime Unit

PCMH - Pleas and Case Management Hearings

Pslist - Print List of Running Processes

Psscan - Scan for EPROCESS Objects

RAM - Random Access Memory

RPM – Read Process Memory

SWGDE - Scientific Working Group on Digital Evidence

SOCA - Serious and Organized Crime Agency

SP2 – Service Pack 2

SP3 – Service Pack 3

TLS - Thread-Local Storage

US – United States

UK – United Kingdom

UTF-16 - Unicode Format (16 Bits)

US-CERT - United States Department of Justice Cyber Crime

UTF-8 - Unicode Format (8 Bits)

VM – Virtual Memory

VIX tools - Virtual Introspection Xen

WPM – Write Process Memory

WinHex – Windows Hexadecimal Bytes

List of Tables

List of Figures

Table of Contents

“Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.”

Chapter 1 Introduction and Motivation

1.1. Introduction

This chapter introduces the problems associated with the analysis of volatile memory to find what user input can be recovered from the volatile memory of Windows Systems. It outlines the objectives of the research reported in this thesis. It also provides a summary of the work that has been conducted to date concerning the identification of the seven most commonly used applications in businesses and the procedure for the extraction of user input from the volatile memory of Windows systems. Finally an overview of the rest of the thesis is presented.

The development of digital forensic investigation techniques has focused mostly on evidence contained within hard disks. But, recently, there has been a great demand for more tools and techniques to be developed for capturing memory images and analysing their content (DFRWS, 2007). However, while there has been much progress with regards to forensic evidence gathering, little has been done on the analysis of the acquired evidence. Limited efforts have been made into formalizing the digital forensic process or even on what user information stored by applications can be recovered. This research idea is motivated by the fact that the physical memory of a computer system may contain information that cannot be found using traditional hard disk forensic investigation techniques in determining what information is pertinent and useful to support the case at hand.

The highly technical nature of digital crimes facilitated a wholly new branch of forensic science called digital forensics. In this new technology, digital forensic scientists collect digital images, preserving the state of the data on a system, and analysing the data produced, transmitted and stored by digital devices.

Currently, digital forensic research focussed on the acquisition and analysis of non-volatile media (Brian & Joe, 2004). Non-volatile media is any media where the information is not lost when the power is disconnected from the device; examples are hard disks and flash drives. Investigation tools such as Forensic Toolkit (FTK) (AccessData, 2008), En-Case (Guidance

Software, 2008) and Columbo Forensic (Columbo, 2010) have been developed to allow a digital forensic investigator to assess the evidence associated with such media.

An example of volatile media would be the RAM of a computer system (Microsoft, 2008). RAM can be thought of as a device that loses the information stored on it when the power to the device is stopped. If the volatile memory of a device can be acquired then any evidence that can be extracted has the potential to enhance digital forensic investigations; although this form of analysis is still considered to be in its infancy (Harlan & Dave, 2007).

Although it is in its infancy, research into volatile memory analysis is tagged as a promising approach in today's digital investigation. A research paper of (Timothy, 2007) discusses the benefits and drawbacks of traditional incident response methods and compared to an augmented model that includes the capture and subsequent analysis of a suspect system's memory.

This research project will capture the volatile memory of a system and extract user information. The aim of digital forensic remains the same, such as to clarify events of the incident and ultimately, identify its perpetrators. This research work will provide a foundation for analyzing captured memory, and provides suggestions for related work in an effort to encourage forward progress in this relatively new area of digital forensics.

1.2. Problem Statement

This research focuses on application level information and volatile memory analysis. This requires information to be gathered from RAM of a computer system. At the time of writing, the field of digital forensics in the analysis of volatile memory of applications is rapidly evolving. Despite having a variety of practical techniques and tools, there is little theoretical basis to support the analysis of any investigation. Thus the development of such a theoretical basis is thought to be an important research problem.

To date, applications have been developed to extract evidence from a system and to present information from that evidence for analysis (Iain, Jon, Theodore, & Andrew, 2008), (Walters & Petroni, 2007), but there has been little attempt at analysing the information that can be obtained;

possibly because of the vast amount of information that is present in RAM of a computer system and possibly, because nobody really knows what to look for.

However, a document for collecting evidence from a running computer (Todd & Henry, 2006) presents a technical and legal primer for the justice community. This documentation was prepared in the United States by SEARCH, the National Consortium for Justice Information and Statistics.

Therefore, it is known that information related to current and closed processes can be acquired and that the user data from those processes can be extracted, but an assessment of that user data has not yet been performed for common applications (AAron, T. Fraser, & William, 2006), (Eoghan, James, & Cameron, 2008). This would give investigators a clearer idea of what to look for when faced with what might be millions of bytes of data that may or may not be relevant (Jason, Ewa, Derek, & Magdalena, 2007). Because of the volatile nature of this evidence it is not yet known whether it is presentable to a court, but it is considered interesting to assess the information that can be found. In the road map of digital forensics, the first Digital Forensic Research Workshop (DFRWS, 2001) stated that:

"What is missing in the digital realm is any real theoretical data about the details of transformations involved in moving from reality to a digitally processed representation. For example, what happens, exactly, to transform an arrangement of ferrous molecules on a disk to a document displayed by a word processor on a computer monitor? What is the mechanism used to record a scene captured by video camera in compressed video data format? Of course, someone knows the mechanisms in both instances, but can we comment on the "correctness" of the processes involved? Trained and certified forensic serologists can comment on the correctness of DNA evidence via explanations that incorporate findings from molecular biology, population genetics, and probability theory. Most analysis in Digital Forensic Science[1] *cannot make similar claims"* (DFRWS, 2001). Since then, remarkable progress has been made in some areas of digital forensics, such as testing of volatile data collection tools (George, 2007), (Betz, 2005),

[1] Currently, there is no universally agreed term for digital forensics. Some authors call it digital forensic science, computer forensics, or forensic computing.

(George & Robert-Jan, 2005), (Gabriela, 2007), (Nicolas, 2008), (Matthew, 2008), specification of data examination (Andreas, 2008), (Andrew, Andrew, Lodovico, Golden, & Vassil, 2008), (Brian & Eugene, 2006) and volatile memory analysis tools (Harlan & Dave, 2007), (AAron, T. Fraser, & William, 2006), (R.B., van Baar, Alink, & Ballegooij, 2008), (Andreas, 2006), (Brian 2005), (Harlan, 2009), (Dan & Wietse, 2005), (Stuart & Jon, 2005), (Brendan, 2008), (AAron, 2008).

However, little effort has been made on the amount of information that can be recovered from only the computer system memory (RAM) while the applications are still running or even if they are closed. The digital forensic community feels the need for accurate forensic data collection, preservation, examination and analysis. This investigative process has become the most important aspect of digital investigation as the extraction of forensically relevant evidence from physical memory can reveal a user's actions and perhaps even suggest their intentions.

Application level information is defined as information which indicates how the user is (or in the case of terminated process, was) using an application. It is expected that application level information will include spreadsheet data, word document text, browser text, email text and any other user stored data related to the application. By extracting and identifying this application level information, a clear picture of the actions carried out by the user on the Windows system can be built and there is the possibility that user actions on the system can be re-created or reconstructed. There is then the possibility that this reconstruction process may illuminate further information.

The process of reconstructing events of user activities on application will determine the events of what happened during the incident. This is a fundamental activity in any digital investigation, because unless the forensic investigator can determine what happened on the applications and how it is linked to the perpetrators, there is simply no basis for determining why it happened and who may have done it.

This work may be useful because of the increased use of social networking, and other, applications where information may not reside or be stored on the local hard disk of the target system. However, it must be present in RAM at some point. In non-digital forensics, investigators can make links between information and perpetrator's actions with common sense

reasoning which is usually sufficient to analyse events of the incident. For example, a fingerprint on the wall can indicate that someone has touched the wall and the unique shape of the papillar lines can be used to identify the person in question.

In digital forensics, the link between information and a perpetrator's actions is more complex. A single keyboard touch can trigger a chain of events inside one or more digital devices that produces the digital evidence. Common sense reasoning is not always sufficient to comprehensively validate the information collected on the digital device because the logic that is used has to be forensically sound with respect to what the user is using the application for.

According to research work of (John, 2007), the problem of "inconsistencies in interpreting digital evidence in complex attacks and validating forensically relevant evidence" is a specific problem to be solved by the digital forensic community. For example, not all evidentiary techniques of validating forensically relevant evidence that were put forward are, or have been, accepted.

In the US, there was a precedent setting case between Daubert versus Merell Dow Pharmaceuticals (Daubert & Merrell Dow Pharmaceuticals, 1993). This case lays out a set of five elements that must be achieved in order for evidence gathered from digital devices to be accepted in the court of law:

- Testing: Whether the theory or technique can be and has been tested.
- Publication: Whether it has been subjected to peer review and publication.
- Error rate: The known or potential error.
- Existence: Do standards and controls exist and are they maintained.
- The general acceptance of the theory in the scientific community.

The digital forensic tools on the market preserve the state of a system or examine a system to find evidence, and after every incident, the universally asked questions are 'what happened?' and 'how did it happen?' (Iain, Jon, Theodore, & Andrew, 2008), (AAron, T. Fraser, & William, 2006), (Jason, Ewa, Derek, & Magdalena, 2007), (Dan & Wietse, 2005), (Brian & Eugene, 2004), (Brian & Eugene, 2005), (Eoghan, 2007), (Eoghan, 2007), (IAAC, 2009), (David, 2001).

But, rarely, is the question asked ‘how has the data been collected?’, ‘who is responsible for the digital activity?’ (Brian & Eugene, 2003); ‘how has it been interpreted?’; ‘how has the resulting interpretation been conveyed to its audience?’ (Brian & Eugene, 2004) and ‘why an object may be evidence?’

Collecting an object and examining its properties is interesting, but for the evidence to be useful we must identify, investigate, validate and reconstruct ‘what caused that object to have those properties?’ According to (Pavel, 2004), event reconstruction methods or event analysis tools are required to examine evidence and to identify why that evidence has particular characteristics.

As discovered from past research many events can occur at a crime scene, including ones that occurred prior to the incident. This has to be understood so that the incident can be fully explored. This all points to a gap in knowledge regarding the process of reconstructing application level information that has been extracted from the physical memory of Windows computer systems. Hence this will be investigated and analysed in this research project. This research contributes to solving the problem of analysing the acquired evidence extracted from these commonly used applications.

The recovery of user input from the volatile memory of applications can be used to determine the amount of relevant information dispersed throughout the memory and also its “lifetime”.

In addition to this, the memory allocated to an application may contain information that can be used to infer what the user is typing on the application, what user have been doing and what user have been using the applications for; the information which may not be visible when using traditional hard disk forensic investigation.

1.3. Classification and taxonomy of application level information

The below describes the classification and taxonomy of user input of application level information:

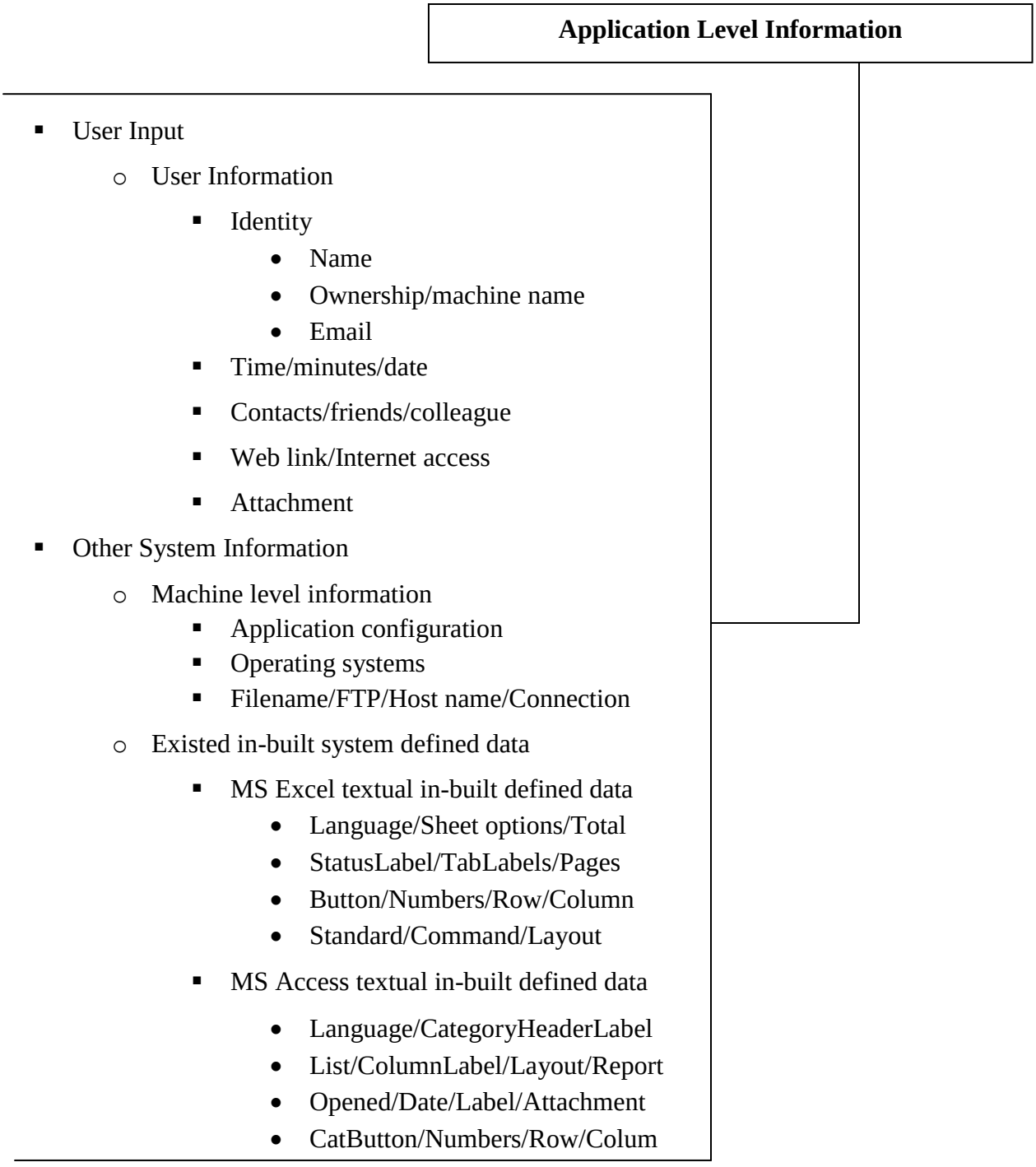


Figure 1.1 Classification and taxonomy of application level information

1.4. Research idea

With the assumptions described above, the investigation of the information contained within the memory of the application may become an essential tool for information assurance as well as solving crime. The aim of this research is:

1. To uncover information that may have previously been "hidden" to forensic investigators by extracting the application level information from the volatile memory of Windows systems.
2. To formalize how the extracted application level information can be reconstructed to describe what user was typing on the application, what user has been doing on the application and what user has been using the application for.

This project required information to be gathered from the RAM of a computer system. Specifically, our method was to capture data and process that data by pattern matching techniques; using the original user input or commonly used English words as the pattern which is matched.

In theory, the investigation on application level information of volatile memory can be carried out as follows:

- First, identify the most commonly used applications to use as the basis for this investigation.
- Second, extract forensically relevant information from the volatile memory that has been allocated to Windows applications.
- Third, validate the extracted application level information to ensure that the information is forensically sound for evidential purposes.
- Fourth, the information will be assessed and used to identify what the user was doing on the application, leading to the development of techniques to reconstruct events that had occurred. This is determining all possible scenarios of the incident that could have happened based on user actions from the state in which the system was discovered.
- Fifth, present the results using quantitative and qualitative assessment techniques.

1.5. Research objectives

The objective of the thesis is to identify and investigate application level information. This will involve completing the following set of tasks.

- To identify the most commonly used applications in Windows computer systems.
- To investigate strategies for strings conversion processes, using the pattern matching technique; the original user input or commonly used English words as the pattern which is matched.
- To develop the approach of reconstructing the extracted application level information.
- To present the user input found on the applications in quantitative and qualitative assessments results that were designed for the research project investigation based on the four scenarios, make recommendations and propose further work.

1.6. Dissertation structure

The structure of the thesis is organised as follows:

- Chapter 2 discusses background and related work in the context of the relevant concepts of forensic science and forensic technology. It starts by giving a literature review of forensic live response and non-volatile and volatile memory forensics. The concept of application level information and volatile memory analysis of Windows systems is also introduced. The legal aspects of forensic technology, digital forensic and digital investigation processes will be discussed. Finally, an overview of previous work defining forensically sound evidence and evidence validation will also be presented.

- Chapter 3 introduces the methodology taken in this research project to investigate the most commonly used applications, the extraction procedures of application level information from Windows systems and the analysis of the information found. It reviews the methodological approaches to the research scenarios, the research strategies and the design, data capturing and procedure for assessing user input. The quantitative and qualitative assessments of application level information are described alongside the need for the theory of application level information in digital forensics.

- Chapter 4 presents the quantitative results and data analysis of the extracted application level information from the volatile memory of Windows applications. It describes some of the user input that is used to present data in the quantitative assessment. In this, four different scenarios of the research tasks are presented and the applicable metric of the investigations are discussed. The chapter presents the key results of the quantitative assessment techniques of the application level information recovered from Windows applications.

- Chapter 5 presents the qualitative results and data analysis of the extracted application level information from the volatile memory of Windows applications. The qualitative assessment is formulated to infer what can be used to describe what the user was typing on the applications, what they have been doing and what they were using the applications for. It describes the reconstruction processes of the application level information found on Windows applications.

- Chapter 6 discusses the results in the context of forensic examiners of the analysis of user input on volatile memory of Windows applications. These are based on the quantitative and qualitative assessment of user input presented in Chapter 4 and in Chapter 5. This includes the scenarios of the research project, the metrics of the experiments carried out and the ability of the tools used. The importance of volatile memory analysis on the theory of application level information will be discussed, including the effect of the memory size allocated to the applications. Following this, the procedure is formulised for use by the forensic investigators and validity of forensically sound application level information is discussed.

- Chapter 7 presents the summary of the theory of application level information and volatile memory analysis of Windows computer systems and concludes with a discussion of possible issues, recommendations and proposals for further work.

1.7. Summary

If the research theory of application level information and volatile memory analysis of Windows applications are to be accepted as evidence information in the court of law, better knowledge of the information about user input applications is needed. This includes a thorough understanding of how the extracted application level information can be used to describe what a user is typing on the applications, what a user has been doing and what a user has been using the application for. The method to be used for searching for data, such as the pattern matching technique developed for finding user input that is stored in the application memory, has to be validated and forensically sound. Certain procedures guiding the use of this technique have to be tested and accepted, as well as its admissibility as a scientific method of gathering digital data. The objectives of the research are to investigate the user input recoverable from the volatile memory allocated to applications.

Chapter 2 Review of Application Level Information Theory and Volatile Memory Analysis of Windows Computer Systems

2.1. Introduction

In this chapter background information related to Windows volatile memory management, volatile memory forensics and the concept of application level information are presented. Previously published literature related to information capture and analysis, the applicable legal aspects of digital forensic technology and digital investigation processes are discussed. Related literature on forensically sound evidence and evidence validation will also be reviewed. In the past, digital forensic research focussed on the acquisition and analysis of non-volatile media (Brian & Joe, 2004). Non-volatile media is any media where the information is not lost when the power is disconnected from the device; examples are hard disks and flash drives. Investigation tools such as En-Case (Guidance Software, 2008), FTK (AccessData, 2008) and Columbo (Columbo, 2010) have been developed to allow a digital forensic investigator to assess the evidence associated with such media. The volatile media of a computer system can be thought of as a device that loses the information stored on it when the power to the device is stopped. An example of volatile media would be the RAM of a computer system.

The RAM of a computer system will be referred to as volatile memory for this research project. If the volatile memory of a computer system can be acquired then any information that can be extracted from that memory has the potential to enhance digital forensic investigations (Jason, Ewa, Derek, & Magdalena, 2007). Volatile memory is a memory area that has both read and writes capabilities (Russinovich & Solomon, 2009). Windows uses the volatile memory available by securing a part to be used by Windows processes and allocating other areas to applications that the user executes. When an application is executed by the user, one or more processes will be started and these processes will be allocated on their own space in the volatile memory. Modules are usually copied into the RAM to be executed and any module loaded from this area will use volatile memory space. Each of these processes may start one or more of their own threads to execute the program code required for the functionality of the application. These threads will use the same space of volatile memory that has been allocated to their parent processes. Therefore, if the volatile memory allocated to a process can be extracted, the

information related to the functionality of the application can then be determined. These findings may aid a digital forensic investigation.

2.2. Windows memory management

Windows divides the volatile memory available to it into two types of address spaces, the process address space and system address space (Russinovich & Solomon, 2009). The process address space consists of the linear address range, this is presented to each process and more importantly, the address within this space is what the process is allowed to use. System address space is the amount of memory allocated for all possible addresses for a computational entity such as a device, a file, a server, or a networked computer. This may refer to a range of either physical or virtual addresses accessible to a processor or reserved for a process. As unique identifiers of single entities, each address specifies an entity's location.

When Windows allocates volatile memory to processes it keeps a track of this allocation in its Process Control Block (Russinovich & Solomon, 2009). This is a data structure in the operating system kernel containing the information needed to manage a particular process; an ID number that identifies the process (Process Identifier), pointers to the locations in the program and its data where processing last occurred, pointers to the upper and lower bounds of the memory required for the process, register contents, states of various flags and switches, a list of files opened by the process, the priority of the process and the status of all I/O devices needed by the process.

Virtual memory is a *logical* version of memory that can completely hides the operating system's management of physical memory (Russinovich & Solomon, 2009). The application's view of memory is simplified when virtual memory is used by using *virtual addresses* to access memory in the operating system. Applications in the operating system are treated as *though they use physical memory, but the Operating System (OS)* can move code and data in the physical memory whenever necessary. Because virtual memory provides a logical view of memory that might not correspond to its physical layout at run time the memory manager, with the assistance from hardware, might translate, or map, the virtual addresses into physical addresses, where the data is actually stored (Russinovich & Solomon, 2009). Figure 2.1 of virtual memory illustrates

that every time an application attempts to access memory using a virtual address, the operating system will secretly translate the virtual address into the physical address where the associated code or data actually reside. However, there is generally insufficient physical memory to contain all processes running simultaneously; the Windows operating system therefore simulates a larger memory space. This is achieved by creating a virtual address space for each process that is translated to physical storage locations through a series of data structures.

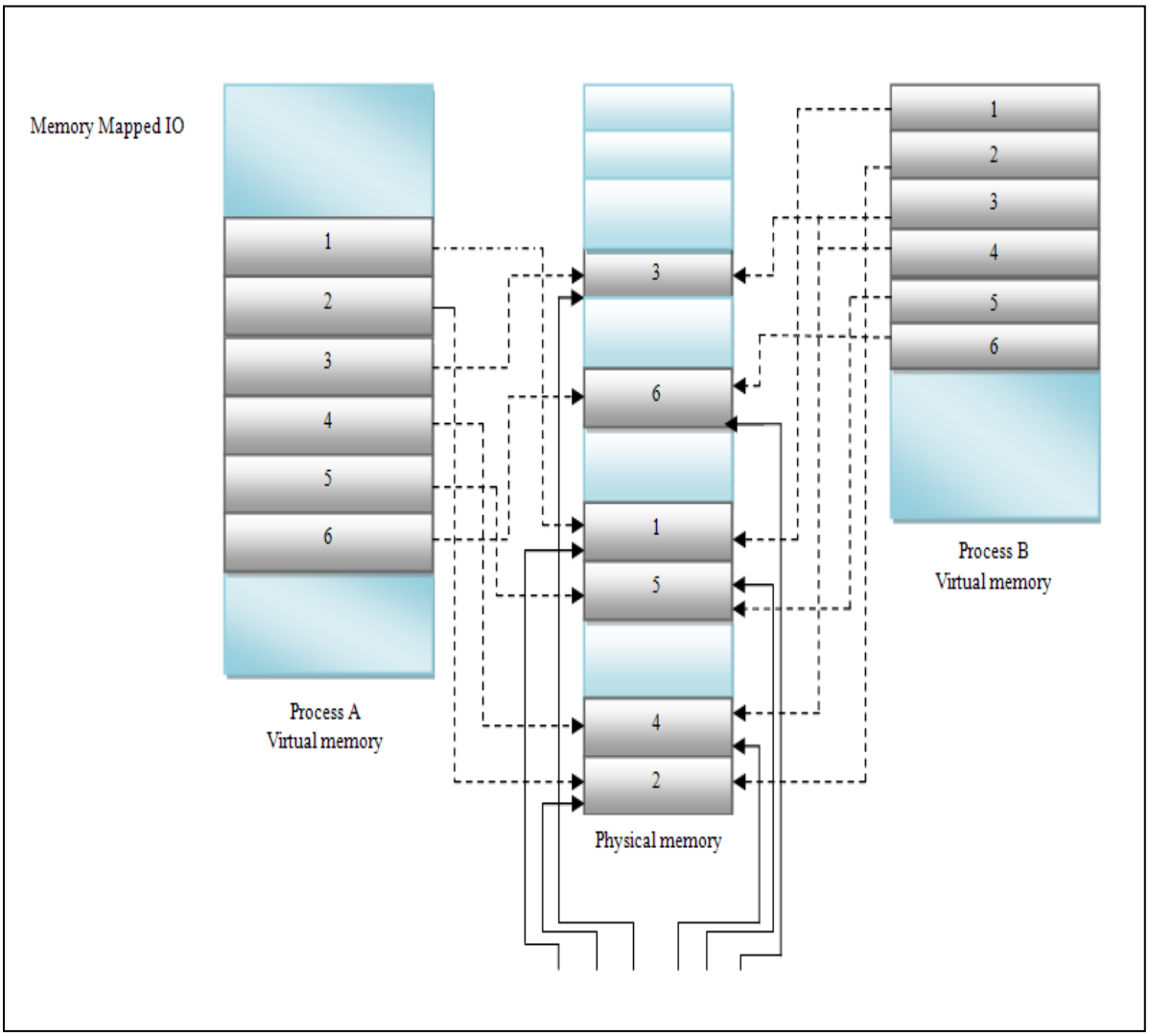


Figure 2.1 Virtual Memory[2]

[2] Taken from Windows internal systems by D.A. Solomon and Russinovich

The fundamental aspect of memory is that the locations of data used by the operating system are not the same as the physical locations needed to locate data in a memory. Thus, by controlling the protection and mapping processes, the operating system will ensure that individual processes do not bump into one another or overwrite operating system data (Russinovich & Solomon, 2009).

Figure 2.2 illustrate three virtually contiguous pages mapped to another three discontiguous pages in physical memory.

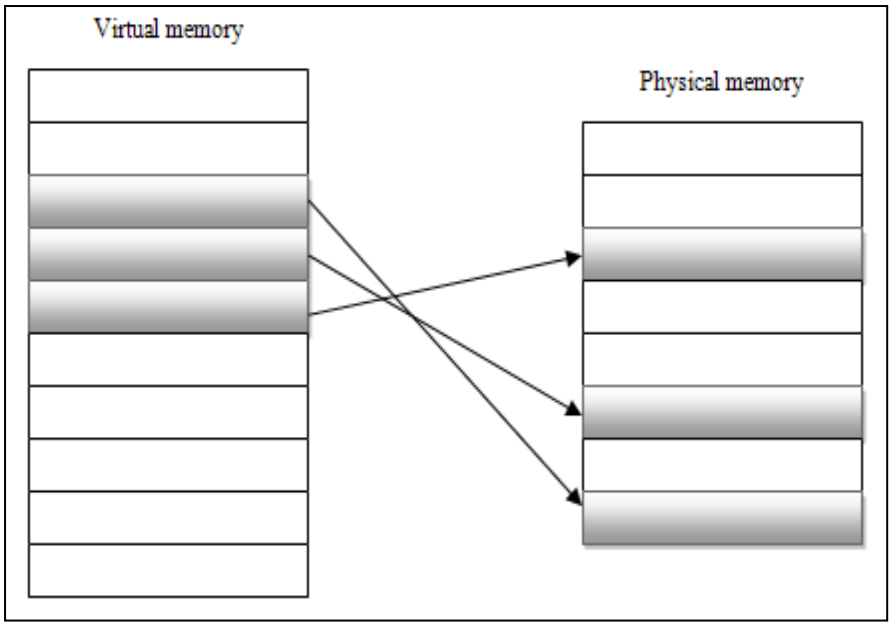


Figure 2.2 Mapping virtual memory to physical memory[3]

2.3. Process level information

When volatile memory is allocated to an application's process, a large amount of information will be stored there which would not be recovered using traditional non-volatile memory forensic techniques. The standard contents of volatile memory that would be allocated to a Windows process consist of the following:

- The executable code, which defines initial code and data, and this is mapped into the process's virtual address space;

[3] Taken from Windows internal systems by D.A. Solomon and Russinovich

- a list of open handles to various system resources, such as semaphore, communication ports, and files, that are accessible to all threads in the process;
- a security context called an access token that identifies the user, security groups, and privileges associated with the process;
- a unique identifier called a process ID;
- any user input to that process such as files opened, text typed or other general usage of the application;
- at least one thread of execution.

It should be noted that this process information will always reside in volatile memory while the process is running (i.e. while the parent application is running). It is also possible to find this process information in volatile memory after the process has ended (either the application killed the process as it was no longer needed or the application has been closed). In order to find this process information in volatile memory after the process has ended, the memory that was allocated to that process should not have been allocated to another process. It is possible to find the process information of closed processes in volatile memory by looking for standard data structures that are associated with processes.

The information found with the allocated address space may directly be related to the functionality of the process. However, if the memory had previously been allocated to a different process then remnants of that different process will exist in the memory space allocated to the second process. This means that there is a possibility that the memory allocated to a process contains information which is not directly related to that process. In the private address space of one or more threads, each process has security identification and a list of open handles to objects such as files, shared memory sections or one of the synchronisation objects such as mutexes, events (Russinovich & Solomon, 2009). Each process has a security context that is stored in an object called an access token. Threads do not have their own access tokens, the process access token contains the security identification and credential for the thread.

Applications do not have to be altered in any way to take advantage of paging because hardware support enables the memory manager to page without the knowledge or assistance of processes

or threads. The volatile memory is used by the application running a process in the operating system. Figure 2.3 illustrate a process and its resources.

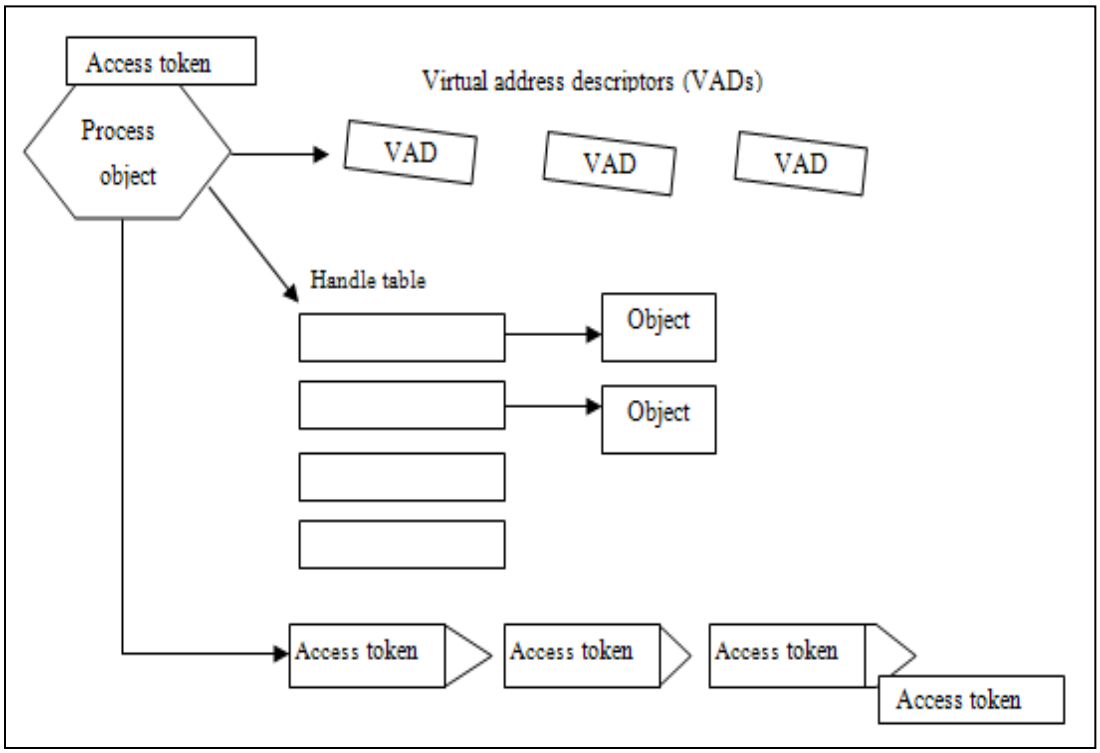


Figure 2.3 A process and its resources[4]

In addition, the data structure associated with the process and thread indicates that a Windows process has its own private memory space; the kernel-mode operating system and device code share a single virtual address space. Each page in virtual memory is tagged as to what access mode the processor must be to read or write the page (Russinovich & Solomon, 2009).

Pages in system space can be accessed only from the kernel mode whereas all pages in the user address space are accessible from user mode. Pages in a process address space are free, reserved, or committed. Using the two step process of reserving and committing memory, we can reduce memory usage by deferring committing pages until needed but keeping the convenience of virtual contiguity. Reserve address space is simply a way for a thread to reserve a range of virtual addresses for future use while, committed pages are pages that, when accessed, ultimately

[4] Taken from Windows internal systems by D.A. Solomon and Russinovich

translate to valid pages in physical memory. Figure 2.4 illustrate the data structure that is associated with process and threads.

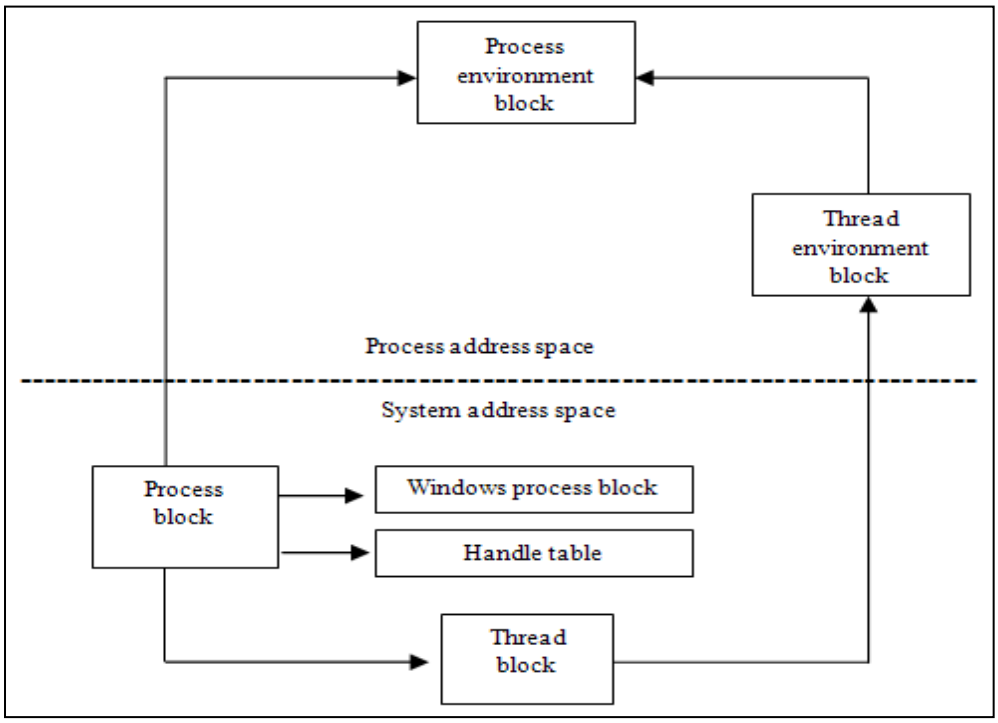


Figure 2.4 Data structures associated with process and threads[5]

Applications can first reserve address space and then commit pages in that address space whereas, memory can be shared to reserving and committed memory. Shared memory can be visible to more than one process or present in more than one process's virtual address space. For example, if two processes use the same DLL, it would make sense to load the referenced code pages for that DLL into physical memory once and share those pages between all processes that map the DLL. But each process would still maintain its private memory areas in which to store private data. Each Windows process is represented by an EPROCESS block, which varies with the version of Windows.

The EPROCESS block contains and points to a number of other related data structures (Russinovich & Solomon, 2009), but because its structure is well defined, instances of

[5] Taken from Windows internal systems by D.A. Solomon and Russinovich

EPROCESS blocks can be found by searching through acquisitions of volatile memory looking for characteristics of the EPROCESS data structure. Figure 2.5 illustrate the EPROCESS block and its related data structure.

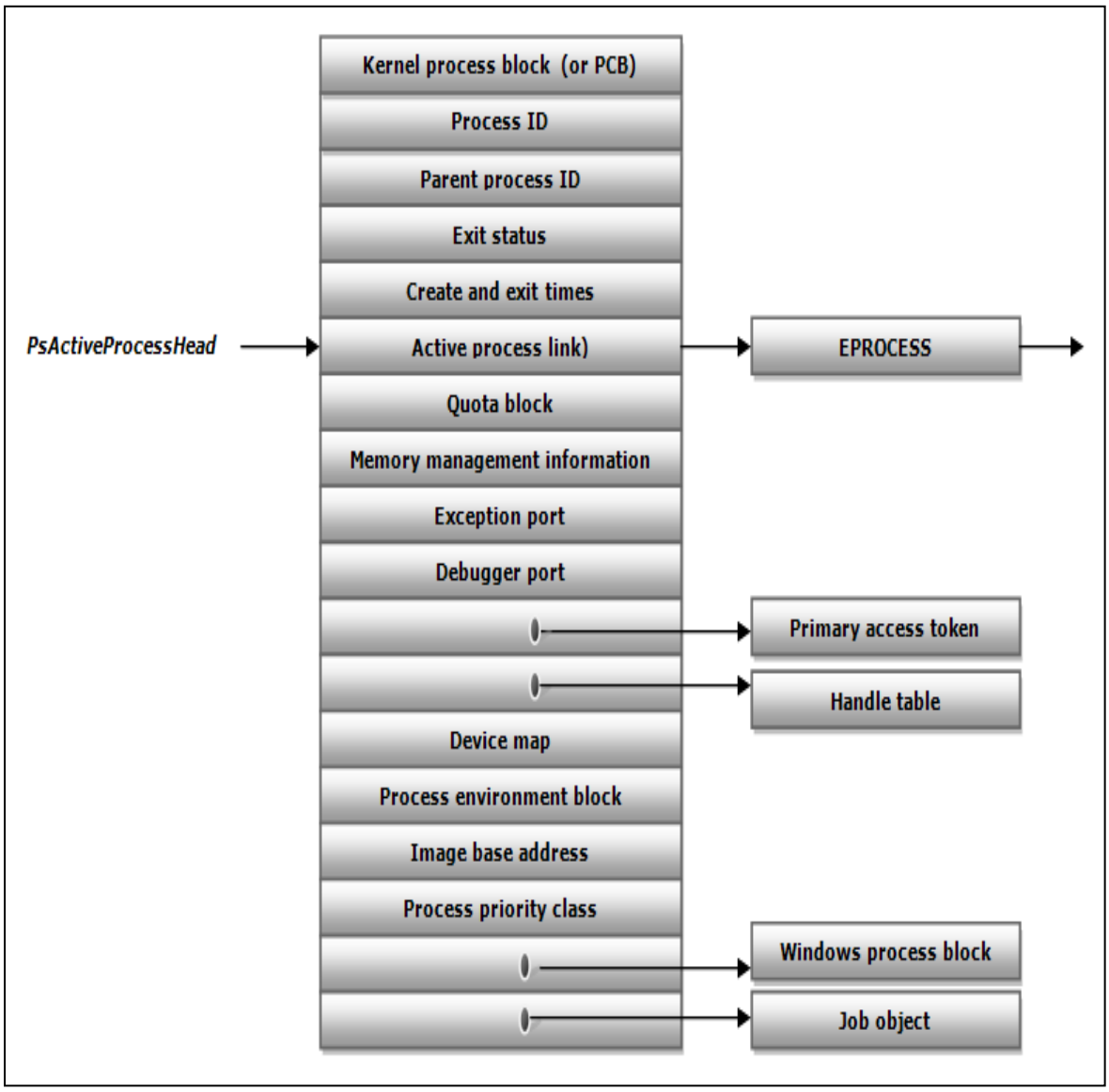


Figure 2.5 Structure of an EPROCESS block[6]

Within a process, a thread exists in Windows and is scheduled for execution. Without a thread, the process's program cannot run. The essential components of a thread includes the content of a

[6] Taken from Windows internal systems by D.A. Solomon and Russinovich

set of CPU registers representing the state of the processor, two stacks - one for the thread to use while executing in kernel mode, one for executing in user mode and a unique identifier called a thread ID (Russinovich & Solomon, 2009). In a thread, a private storage area called thread-local storage (TLS) is present for use by subsystems, run-time libraries and DLLs. Threads sometime have their own security context that is often used by multithreaded server applications that impersonate the security context of the clients that they serve. The volatile register, stacks and private storage area are called thread's context. This information is different for each of the Windows machine architecture and because of this structure, by necessity; it is termed architecture-specific (Russinovich & Solomon, 2009). Although threads have their own execution context, every thread within a process shares the process's virtual address space, that is, all the threads in a process can write to and read from each other's memory.

2.4. Volatile memory forensics

The analysis of volatile system data presents significant challenges and risks (Garcia, 2007). In digital forensics the examiner cannot examine volatile system data without making some change to it, however minor. From the moment the investigator logs on to the target system, logs are recorded, temporary files are created and deleted, network connections can be opened and closed, history files are updated, and registry entries are queried, either added and or modified. All of these activities change the state of the system, and as such may contaminate the evidence that the investigator hope to collect. Depending on the tools and techniques used in memory acquisitions, these changes may affect the data collected. Any evidence discovered cannot be recorded on the target system without further system modification, so where necessary, a network connection is often used to pipe data to another system under the investigator's control, where the results of the volatile data analysis can be recorded. It is also unlikely that a target system contains all of the utilities required to perform a forensic analysis, so some additional media loaded with analysis tools may be attached to the system, resulting in more system activity that can potentially contaminate or overwrite essential digital evidence.

There are two ways of acquiring volatile memory data from Windows systems. This includes the hardware-based memory acquisition method and software-based tools. The method presented in (Garcia, 2007) is among the few hardware-based memory acquisition methods that change

memory contents as little as possible by using a PCI extension card to dump the memory content to an external device. A range of software-based tools have been recently developed for memory acquisition and memory analysis. With regards to memory acquisition, (Msuiche, 2008) is a command line tool that reconstructs the virtual address space of the system process and other processes. A method of (ManTech, 2008) is a tool that is capable of revealing hidden and terminated processes and threads. Win32dd (Frederick, 2006) and Nigilant32 (Matthew, 2010) are tools that can capture the physical memory of computer systems. In addition to these tools, MemParser (Betz, 2005) and the Volatility Framework (AAron, 2009) are examples of other tools that can perform memory analysis. Of these two, the Volatility Framework is more extensive. This tool is capable of performing the analysis on a variety of memory image formats such as DD format, crash dump and Hibernate Dumps. Volatility is able to list OS kernel modules, drivers, open network socket, loaded DLL modules, heaps, stacks and open files. The research work of (Carvey, 2007) addresses the need for more sophisticated tools on physical memory acquisition and analysis. This is data carving method which is a recovery approach that is frequently used during digital investigations. Moreover, it is essential that a new development tools should integrate different approaches. A new model of (Stefan, Tobias, & Jana, 2009), point towards the graphics extraction that is contained in a memory dump.

An investigation into the case relevance of volatile information was performed by (Ruibin, Tony, & Mathias, 2005). This research is focused on computer intelligence, the approach is related to the current computer forensic frameworks, such as "FatKit" that was presented by (Walters, Fraser, Petroni, & Arbaugh, 2007). This tool is simply an investigatory assistance procedure that can be reused for sharing information in computer forensics. However, this approach has not reached its full potential because it is limited to the field of case matching in volatile data presentation in the court. The research work of the empirical studies of information retrieved from volatile memory and the Fatkit framework of digital investigations presents clear guidance on the legal aspect of scientifically proven technology upon which law can be practised.

Scientifically proven technology is the use of scientific technology in the court of law; it becomes proven when it undergoes the Daubert Tests of (1993) guideline. This guidance is based on the tools, techniques, investigations, analysis and presentation of evidence by expert

witnesses in the court following scientifically proven technology solutions of (Gary, 2010), (Frederick, 2008), (Mark, Reith, Clint, & Gregg, 2002), (Brian, 2005). In the United States, there is a case of Daubert v Merrell Pharmaceutical (1993) whereby clear guidance is given for the use of scientifically proven technology of evidence in the court of law. "*Daubert versus Merrell Dow Pharmaceuticals*" is a landmark case that defined what is required for information to be considered as scientific evidence in a US court of law (Daubert vs. Merrell Dow Pharmaceuticals., 1993). These are the five *"Daubert Tests"* which the scientific method of gathering evidence on digital devices must pass.

A research work of (Frederick, 2008), focuses on the fundamentals of digital forensic evidence and lay emphasis on the fact that digital forensic evidence should be considered in the light of the legal context of the matter at hand. An examination into digital forensic by (Mark, Reith, Clint, & Gregg, 2002) stated that law enforcement agencies are in a perpetual race with criminals in the application of digital technologies and therefore, a demand for the development of tools to systematically search digital devices for pertinent evidence is required. A paper of (Brian, 2005), developed a digital investigation tool because of the demand in forensic analysis of digital crime scene investigations. This tool is a collection of file system and disk images that test the functionality of analysis tools. Information contained in applications can give investigators a clearer idea of what to look for when faced with what might be millions or billions of bytes of data that may or may not be relevant (Brian & Eugene, 2006) to crime investigations. The Digital Forensic Research Workshop (Palmer, 2001) highlighted a need to research on the use of information obtainable from volatile memory for evidentiary purposes.

2.4.1. Digital investigation process

The U.S Department of Justice published a process model in the Electronic Crime Scene Investigation as a guide to first responders which consists of four phases; collection, examination, analysis and reporting (NIJ, 2001). The analysis phase of this model is improperly defined and ambiguous. Research to develop a road map of digital forensics (presented at the Digital Forensic Research Workshop (DFRWS, 2005)**,** indicated the need for the digital investigation process and presented a general standard of techniques. This investigation process

is too cumbersome because of the stages of the analysis to be followed which is not specific to the research purpose of volatile memory.

According to (Siti, Robiah, & Shahrin, 2008), a research work of mapping process of digital forensic investigation framework presents three main issues for forensic investigation process. The three main issues were analyzed from the frameworks, which are process redundancies, area focus and framework characteristics. The other existing digital investigation process framework was reviewed. For example, the digital investigation framework of the research work of (Mark, Mark, Clint, & Gregg, 2002) and (Baryamureeba & Tushabe, 2004) focussed on duplication process of digital investigation including, the activities of digital investigation and incident response in their framework while the research work of (Brian & Eugene, 2003) and (Marcus, James, Rick, Timothy, & Steve, 2006) focussed on building a method for more rapid forensic digital examinations of incidents whereas the research work of (Stephenson, 2003), (Nicole & Jan, 2005) and (Felix & Bastian, 2007) for digital investigative framework were based on the analysis process of evidence to be obtained from the investigation and therefore presents an improved overall process of digital investigation. Amongst these frameworks, the research work of (Nicole & Jan, 2005) and (Marcus, James, Rick, Timothy, & Steve, 2006) gives the characteristics and the practical aspect of digital forensic process as specific and important to the investigations process. A research of (Brian & Eugene, 2003), is focused on the examination of digital forensic models. This model was presented as specific to an integrated digital crime scene investigation process.

The research work of (Carrier, 2006), presents hypothesis-based approach to digital forensic investigations and the forensics process proposed by (Karen, Suzanne, Tim, & Hung, 2006) consists of four phases which are collection, examination, analysis and reporting. The output of this forensic process for each phase is similar to the early process proposed by (Pollitt, 1995) which transforms media into evidence either for law enforcement or an organization's internal usage. First, the transformation occurs when the collected data are examined. By this process, data from media will be extracted and be transformed into a format that can be processed by forensic tools. This data will of course, be transformed into information through analysis and finally, the information will be transformed into evidence during the reporting phase. A paper of

(Felix & Bastian, 2007) presents a common model for both incident response and computer forensic processes which combine the concept in a flexible way to improve the overall process of investigation. This process of investigation allows for a management oriented approach in digital investigations while retaining the possibility of a rigorous forensic investigation. This framework focused greatly on the analysis of data collected from digital investigation and it consists of pre-Incident preparation, pre-analysis, analysis and post-analysis. All of these frameworks have their own strength however, there is no single framework that can be used as a general guideline for investigating all incidents or crime related cases (Brian, 2005).

A research of (Brian & Eugene, 2003), on the examination of digital forensic models presented a model that is specific to an integrated digital crime scene investigation process. This model includes six major stages, preservation of digital scene, survey for digital evidence, document evidence and scene, search for digital evidence, digital crime scene reconstruction, presentation of digital scene theory. According to (Brian, 2005)**,** there is no single way to conduct an investigation but, the combination of the investigation approach of (Brian & Eugene, 2003) and (Felix & Bastian, 2007) was found suitable in this research project. Figure 2.6 shows the four major phases of the digital investigation process of application level information.

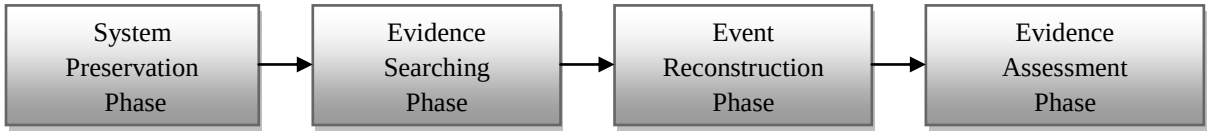


Figure 2.6: The four major phases of digital investigation process of application level information

2.4.1.1. System preservation phase

This preservation phase is similar to the research work of (Brian & Eugene, 2003). It preserves the state of the digital crime scene whereby certain actions are taken depending on the legal, business and operational requirements of the investigation. For example, a legal requirement may involve an investigator making a full copy of the computer hard drive or volatile memory.

2.4.1.2. Evidence searching phase

In this evidence searching phase, user information is searched for on the digital device. This process occurs after acquired memory data has been taken (Brian & Eugene, 2003). This information is preserved for the purpose of looking for data that supports the hypothesis about the incident.

2.4.1.3. Event reconstruction phase

This is the reconstruction stage whereby the forensic investigators use the information that has been found to determine what digital events occurred in the system. According to (Brian & Eugene, 2003), gathering evidence from digital devices may require the use of different techniques but reconstructing the user activities from digital evidence requires special technical skills including knowledge about the applications and the operating system (OS) that are installed on the system. Once events related to the incident have been reconstructed (Brian & Eugene, 2004), the investigator may be able to correlate the digital events with physical events.

2.4.1.4. Evidence assessment phase

This is the last stage of the investigation whereby the forensic investigator analyses the information that has been found on the applications to determine the information that can be used as evidence. A research work of (Felix & Bastian, 2007), focused greatly on the analysis of data collected from digital investigation and it consists of pre-incident preparation, pre-analysis, analysis and post-analysis. The information analysed can be presented for evidential purposes in the court of law.

2.4.2. Volatile memory acquisition

Applications have been developed to extract information from the volatile media of a system in order to analyse it (Carrier, 2006), (Cameron, Eoghan, & James, 2007). A paper of (Brendan, Abhinav, Patrick, & Jonathon, 2009) described an automated technique for generating robust signatures for kernel data structures. These processes run actively in the operating system. This research work presents a methodology to derive signature for EPROCESS blocks and the data structures used to represent a running processes, but this work is limited to existing signatures used by memory.

A paper of (Brian & Kara, 2008) investigated how volatile memory is used by the operating system. This investigation provides a discussion of volatile data in digital forensics analysis through virtual introspection and hence, presents a suite of virtual introspection tools which was developed for Xen (VIX tools). The virtual introspection is the approach of inspecting a machine from the outside for the purpose of analyzing the software running inside it. The VIX tools suite can be used for unobtrusive digital forensic examination of volatile system data in virtual machines, and addresses a key research area identified in the virtualization of digital forensics. This work laid emphasis on the ubiquitous and complex digital forensic techniques and the challenges faced by forensic investigators during incident response.

2.4.3. Volatile memory analysis

There has been little attempt at actually analysing the information that can be obtained from volatile memory; possibly because of the vast amount of information that is present in the RAM (a commonly used volatile media) of a computer system and possibly because nobody really knows what to look for (Harlan, Dave, & Jesse, 2007), (Iain, Jon, Theodore, & Andrew, 2008). Although it is known that information related to current and closed processes on computer system can be acquired, and that the user data from those processes can be extracted, an assessment of that user data has not yet been performed (AAron & Nick, 2007). A research paper (Chris & Kevin, 2003) of incident response and computer forensics pointed out the typical live response information that an investigator may be looking for on the Windows systems memory. This include the system data and time, logged on users and their authorization credentials and network information, connections, and status, process information, memory and process-to-port mappings, clipboard contents, command history, services, driver information, open files and registry keys as well as hard disk images. Various research into these pieces of information have been attended to when attempting to reconstruct the events leading up to a cybercrime activities and security incident. For example, a paper (James & Gilbert, 2010) discusses that the method for collecting memory should not affect the operating system, if no collection method has been implemented a priori, options are limited to arbitrary memory dump and extract process, registry, services, driver information and network communication information.

The research paper of (Brendan, 2008) discusses the forensic analysis of the Windows registry in memory and present the tools and techniques that can be used to extract this data directly from memory dumps. This research is limited to the cached version of the registry and that malicious modifications are easily detectable by examining memory. A research work of (Richard & Eoghan, 2010), focuses on the method of extracting user-entered data retained in Windows applications but limited to Windows command line prompt or details and command history from physical memory. Another research paper of (Okolica & Gilbert, 2011), describes the method of extracting the windows clipboard from physical memory. This research is limited to Windows clipboard structure and the process of retrieving copy/paste information from Windows system.

One potentially an important piece of information is the process information of an application memory and process to port mapping but there is no assessment of user input stored on process information. This area of research of volatile memory analysis is interesting because there is little or no theoretical research study in this area. Hence a research gap is identified. Specifically, this research project is focused on the process information of an application memory to investigate the user input that can be recovered or stored on an application when an application is closed or running on Windows system.

In order to acquire a volatile memory sample an investigator would first establish a trusted command shell and then they would establish a data collection system (Harlan, Eoghan, & Cameron, 2007). The aim of volatile memory analysis is to give the forensic examiner the chance to collect volatile evidence in a 'human-readable' format that is easier to peruse than when it is stored as a binary version (Stuart, Jon, & Suzanne, 2007). This form of memory analysis is vital because when the system is shutdown, evidence in volatile memory might be lost (Brian, 2006). However, this evidence might be useful in analyzing what has occurred at the crime scene and also, reconstructing events leading up to the crime (Joshua, Pavel, Mohd, & Yuandong, 2009). Due to the volatile nature of running memory, the imaging process is taking a snapshot of a 'moving target' (Harvey, Dave, & Jesse, 2007). This may lead to data inaccuracy but the reconstruction of user events makes this research topic an area of interest. The extracted information from the volatile memory allocated to the applications in Windows computer systems will be referred to as Application Level Information.

2.5. Legal considerations

Since, 1782 English law related to the calling of expert witness in the court of law has been in existence (John, 1984). This is the first case in which parties called their own expert witness. The use of experts called by the court for evidence presentation may go back as far as 1345 (Peter, 2011). In support of the expert witness requirement for courts, research on the legal perceptions of science and expert knowledge by (Joseph, Shari, & Neil, 2002) stated that expert testimony has played an increasingly important role in the past half century. In 1784, forensic science was being deployed on physical matching in a murder case.

In the UK, computer forensic products started to appear in the market in the late 1980s and as far back in the 1985, a dedicated computer crime unit was founded. The first good practice guide for computer-based evidence from the Association of Chief Police Officers (ACPO) (ACPO, 1998) came into existence since 1998. In 2006, the UK government disbanded the National Hi-tech Crime Unit (NHTCU) in opting to deal with organised and serious crime through the serious and organised crime agency (SOCA). Therefore, ACPO proposals work hand in hand with the central E-Crime Unit (PCEu) which also work alongside the National Fraud Reporting Centre (NFRC). Over the years, there have been no shortages of initiatives to control, regulate, certify or accredit expertise witnesses in the court based on traditional hard disk forensic evidence presentation, but this approach is limited with regards to volatile memory evidence and digital investigation processes.

The conference proceeding of DFRWS on the road map to digital forensics (DFRWS, 2001) is the first digital forensic research conference and later, in 2005, challenges into volatile memory forensic investigation were introduced (DFRWS, 2005). A paper of (Aaron & Nick, 2007) introduces research tools and techniques for alternative ways of gaining access to information that reside on RAM. Several efforts into the research theory of digital forensic tools and techniques have been made. Recently, Windows Forensic Analysis (Haelan, Dave, & Jesse, 2007) presented digital investigation approaches on incident response and cybercrime investigations.

The UK legal framework of expert witness is required to be specific to the precise circumstances within the jurisdictions in the UK. These jurisdictions include England and Wales taken together, Scotland as separate entity and Northern Ireland. However, the basic rule of law is very similar (Peter, 2011) throughout the UK. English law makes a distinction between technical evidence and expert evidence. According to (Law, 2009), an expert witness gives evidence based on experience and opinion, a criteria which was adopted in England but it was originally set out in an Australian case. Hence, technical evidence was described in (Peter, 2011) as a technical investigation or procedure that should be followed when reporting evidence by an expert witness without commenting on the findings. This process contains the identification of user activities on digital devices as required in any digital investigation. However, the legal procedures of presenting this evidence in court are demanding. Volatile memory contains information that cannot be found in a traditional digital forensic investigation (Peterson & Shenoi, 2009).

The theory of digital forensics in the court of law can be framed by some applicable elements of digital evidence. A research paper of (Erin, M.F.S, & J.D, 2005) presented the confluence of digital evidence and the law. This is the forensic soundness of live-remote digital evidence collection. The research theory is limited to the fact that the acceptance of digital evidence is relevant in a large number of cases and its usefulness in court is subject to Judges' experience, belief and understanding.

The article of (Orin, 2005) indicated the appreciation of digital evidence in the court of law. The information contained in it relates to the familiarity of terminology used; an understanding of both the technology and the language used in reporting digital evidence. Van Buskirk et al (Eric & Vincent, 2006), discuss the challenge of proving the accuracy and reliability of digital evidence. This is intensified because of the variability in forensic software errors in the imaging process but the research theory is limited to the differences in examiners' knowledge that could affect the reliability, accuracy, and integrity of digital evidence.

The research of (Rolf & Ruedi, 2003) indicated a perception amongst the legal community for digital evidence to be accepted and admitted in the court. The evidence must be reliable and correct. Given the importance of digital evidence in criminal and civil court cases, studies have

started to emerge on the use and perception of digital evidence by various participants in the judicial system. A paper of (Michael, Julia, & Marc, 2006) showed that most judges did not hold e-mail and Web site related evidence as valid in their courtrooms. This indicates that judges receive minimal training related to any type of digital evidence. The importance of witness testimony in the court of law particularly that of experts, is expected to be wholly consistent with the Daubert (Daubert v. Merrell Dow Pharmaceuticals, 1993) five tests rule.

A research of (Stephen, 2008) presents the authentication of the data source of internet-based maps that can be used as digital evidence in the court of law but there are judges bias to internet-based maps because of the limited technical knowledge of using it. For example, Google Earth, Google Maps or MapQuest poses a challenge regarding digital presentation in the court. Judges might be skeptical about the accuracy of the source of evidence if he or she has a limited knowledge on the forensic digital technology. Therefore, the authentication of this type of evidence is based on Judges' mindset of technology; their understanding, education and training (Gary, 2010). A typical example includes Facebook or other social networking applications. Judges knowledge and experience about these types of social networking applications in digital evidence are limited because some Judges did not personally use them (Peter, 2009). Therefore, the perception of Judges to digital evidence must be corrected in order for digital evidence to be used in a court. The information on user activities on Windows applications systems may be used as evidence in a court of law but it is not yet accepted in the court of law.

2.5.1. Evidence validation/forensically sound evidence

One of the difficulties in presenting information obtained from volatile memory as evidence is that of Judges' awareness and understanding of the application of digital forensic tools and techniques (Gary, 2010). In recent times, there have been a research theory of (NIST, 2009) which presents forensic strings specification. A document was then issued by the National Institute of Justice (NIJ, 2008) on electronic crime scene investigation and this document provided guidelines for forensic investigators.

The guidelines are as stated below:

- Ensure the officer safety and the safety of others to remain the highest priority.

- Recognize the investigative value of digital evidence.
- Assess the available resources.
- Identify the equipment and supplies that should be taken to electronic crime scenes.
- Assess the crime scene and the digital evidence present.
- Designate the assignments, roles, and responsibilities of personnel involved in the investigation.

A recent paper of (Peter, 2011), discusses quality standards, steps to certification and accreditation, as well as the assessment of digital forensic by expert witnesses in the court of law.

The quality standards are highlighted by Home Office UK (Home Office UK, 2010) and include the following:

- Quality standards applying to organizations and scientific processes;
- guidelines for validating new developments in forensic science;
- competence standards applying to individual forensic practitioners.

This document indicates the UK solution to digital forensic standards and procedures in court proceedings which are governed by the Forensic Science Regulator. This includes a Forensic Science Advisory Council in which specialist group was formed in 2010. The research theory of (Peter, 2011) discusses the assessment of digital forensic by expert witnesses in the court of law. The stages of assessment that were presented by (Home Office UK, 2010) are as follows:

- the needs of law enforcement to obtain good quality forensic work in the "technical" as opposed to "opinion" category are addressed by the current schemes of the Forensic Science Regulator.
- the Legal Services Commission, as the provider of publicly funded legal aid already receives copious documentation about experts, their reports, their works-sheets, and notice from the courts if work has been unsatisfactory.
- in terms of the point at which a judge has to decide whether to allow someone to give opinion evidence, there is already increasing use of case management powers through the mechanism of pre-trial hearings, Pleas and Case Management Hearings (PCMH) and

there seems no reason why more time should not be expended on the issue of experts to determine actual experience and the significance of specific claimed qualifications

- if the Law Commission's proposals to develop a Daubert series of tests on the general acceptability of an item of novel scientific or technical evidence are accepted and passed into law, judges will be able to exercise further control.

A research of (Warren, 2008), an empirical study of information retrieved from volatile memory presented a measure of how much more likely an hypotheses is believed to hold after evidence is considered. This measure is applied to analyzing evidence in the form of coordination level and inverse document frequencies. This measure contains specific information to be analysed for a specific relevance which can be used for data retrieval and for data accuracy.

A paper of (Butler, 2007), validated forensic applications for volatile memory acquisitions and several questions were asked on what it is, why does it matter and how should it be done. Forensic validation builds confidence for the court as well as aiding quality assurance in the lab but there is yet no standardized strategy for memory analysis that is generally accepted in the court.

2.6. Application level information

In this research project, application level information is defined as the extracted user input from the volatile memory of Windows systems. This user input is stored over time and dispersed in the volatile memory of these Windows applications by the Windows operating system. The research intention on application level information was as a result of the limited research on digital forensic tools and techniques for volatile memory analysis. For example a paper of (Brian & Eugene, 2006) presented a research statement that information contained in common applications can give investigators a clearer idea of what to look for. This information may include what user is typing, what they have been doing on the computer and what user are using the application for.

However, the physical memory contains millions of bytes of data that may or may not be relevant to crime investigations. A research work of (Andrew, Andrew, Lodavico, Golden, &

Vassil, 2008), presented research tools for memory analysis and give attention to the analysis of memory stack residues, a technique and tool for searching processes and threads in windows memory dumps. This tool is most closely related to Pyflag (Michael & David, 2007), which allows viewing of log files, network traces, Windows memory dumps (by incorporating Volatility), and other data within a common framework. But, the research tool of (Brian & Eugene, 2006) is differentiated because of its correlation capabilities from a variety of forensic targets which demonstrates the integrated analysis and correlation of disk image, memory image, network capture, and configuration log files. Therefore, the research into application level information takes a different approach of scenario-based investigations of user input on commonly used Windows applications. The research approach will detail information on a possible case scenario of volatile memory acquisition and application memory analysis when the user input are stored on an application hence, the application may be closed or running on Windows system.

A recently published paper (Schuster, 2006), indicate the use of Volatility tool to list kernel modules that resides in operating system including, loaded DLL modules, drivers, open network socket, heaps, stacks and open files. A paper of (Iain, Jon, Theodore, & Andrew, 2008) reviewed several tools and techniques for acquiring volatile data from the operating system of Windows computer systems including the evidence collection from live data analysis. A research study of (Peter, 2011) focused on its classification and evidence admissibility in the court and thus emphasized that volatile memory analysis process is too immature and the technique of investigations are not currently applicable in a court of law in the UK.

A theory of (IAAC, 2007), argued that where trusted validated tools are deployed, the reported results could be considered as: "*testimonial evidence (Testimonial – the eyewitness observations of someone who has present and whose recollections can be tested before the court*"). In this perspective, data collected by forensic investigators must be performed in such a way that it is legally admissible in court cases. A research discussion at (DFRWS) reinstated the importance of digital evidence in the court (David, 2001). The issues related to Judges awareness was presented by (Gary, 2010). Understanding of the application of digital forensic in the court of law was discussed as necessity by (Peter, 2011).

A research work of (James, Paul, Jessica, & Joseph, 2010) presented research on electronic discovery of evidence chain of custody and control, it was said that for evidence to be forensically sound, the manner used to obtain the evidence must be documented.

A good practice for abstract level of string searching evidence was produced by (NIST, 2009). This research makes clear the state of something to search with, some place to search, something to search for and the required search results. At the moment, the information collated from live data presents some admissibility difficulties in the court (Iain, Jon, Theodore, & Andrew, 2008) but this information can reveal vital information to investigations.

A paper of (Hejazi, Talhi, & Debbabi, 2009) argued for the support of digital evidence in the court of law and carried out investigations on the extraction of forensically sensitive information from Windows physical memory but this research does not assess the user entered information on applications. The research is limited to looking at call stacks for sensitive information like passwords.

A research paper of (Richard & Eoghan, 2010) focused on the extraction of Windows command line information from physical memory, this research is limited to a command history and commands history elements. The researcher only presented the need for memory forensic techniques to extract user-entered data that is retained on various Microsoft Windows applications. Therefore there is need for the formalization of application level information.

2.6.1. Formalization of application level information

The digital forensic investigation of volatile memory is a relatively new discipline. Many of the existing laws that are used to prosecute computer related crimes establish precedents with relation to computer forensics. There is need for a new court ruling on the use of volatile memory analysis. In the UK, evidence from volatile memory has not yet been accepted in a court. But as shown by the United States Department of Justice Cyber Crime (US-CERT, 2008), (Richard, Colin, Jake, & Cal, 2005), some States in America have accepted the use of volatile memory as an evidential source of information. As a result of this information, the importance and usefulness of volatile memory analysis in the court of law has become very important. The research project is focused on the extracted application level information from the physical memory of Windows computer systems. As earlier stated, application level information can be

described as relevant user information that is stored and dispersed in the physical memory. It is expected that it can be recovered and extracted for evidential purposes. This information can be recovered while the application is still running or closed on a Windows system. The relevant information may be related to what user is typing, what they have been doing and what user is using the applications for on the Windows system. The time aspect of relevant user input can therefore be determined to ascertain how much information can be lost on these applications.

2.6.2. Extraction and analysis of application level information

A paper of FATKiT (Walters, Fraser, Petroni, & Arbaugh, 2007) described a framework for the extraction and analysis of digital forensic data from volatile system memory. FatKit enables the automation of the extraction of objects from memory. However the limitations of FatKit are that it is used for a static memory dump file analysis and in a real mode it has a 4GB segment limit. It was said that digital forensic of volatile memory requires expert skills because data acquisitions and volatile memory analysis are still at its infancy. A paper of (Mariusz, 2007) investigated Windows memory dumps and searched for processes and threads. This method uses a PCI extension card to dump the memory content to an external device, but this approach is limited to hidden and terminated processes and threads.

A recently published paper of (James & Gillbert, 2011), discusses the method of extracting the clipboard from Windows physical memory. This paper presented the structure and the process of retrieving copied information from the clipboard. This research work is limited to copying of last file used by user or the last password used. This provides investigators with invaluable information during forensic investigations. Upon the above mentioned theories, the research gap was identified for the extraction and analysis of the application level information. This research therefore focused on how related user activities can be recovered from the physical memory of Windows systems. Therefore the forensic digital investigation includes the extraction of application level information from the volatile memory of Windows applications which may be used to solve crimes. The research further investigates the quantity of information stored on the physical memory and also, carries out investigations on the quality information that can be recovered from the volatile memory allocated to the applications.

2.7. Validity of forensically sound application level information

The National Institute of Standards and Technology (NIST) is one of the pioneers in pursuing the validation and verification of computer forensic tools. Within NIST, the Computer Forensics Tool Testing (CFTT) project (NIST, 2008) was established to test digital forensic tools. The activities conducted in forensic investigations are separated into discrete functions, such as string searching and a test methodology is developed for each category. The Computer Forensic Reference Data Sets (CFReDS**)** was developed by National Institute of Standards and Technology (NIST) and a typical scenario of strings searching on container and nested container files was presented (NIST, 2010). The documentation criteria for the testing of digital investigation tools were provided by Computer Forensics Tool Testing (CFTT) project. Although, there has been little public testing on what information can be extracted from applications, Carrier (Brian, 2005) has developed small test cases, called Digital Forensics Tool Testing Images (DFTT). A review of these strings searching criteria indicates several functionalities and tools that can be used for the extraction of strings from the application memory.

In addition, it is necessary to investigate into the development of application level information as a scientific method of gathering data and to ascertain if it can be used as evidence in a court of law. In the United States, there is a case of Daubert v Merrell Pharmaceutical (1993) where clear guidance is given for the use of scientifically proven technology of evidence in the court of law. "*Daubert versus Merrell Dow Pharmaceuticals*" is a landmark case that defined what is required for information to be considered as scientific evidence in a US court of law (Daubert v. Merrell Dow Pharmaceuticals., 1993). There are five *"Daubert Tests"* regarding the admissibility of scientific examinations and for their use as a method of gathering evidence on digital devices. These Daubert tests include, first, the testing of the technique used, for example, the pattern searching techniques of the research project have been tested and validated, second, the peer review of the technique through research publications, for example, the number of research papers published from this research project, third, the potential error rate of known and unknown, for example the user input found in the application memory using the two pattern matching techniques as demonstrated in Chapter 4, fourth, the standards controlling the use of the technique and fifth, general acceptance of the theory in the scientific community.

2.7. Summary

If application level information is to be accepted in the court of law in the UK, better knowledge of forensic technology tools and techniques is required. This includes a thorough understanding of the digital investigation processes, the volatile memory acquisition techniques and the adopted volatile memory analysis tools. This has to be forensically sound according to the five *Daubert* tests. The application level information has to be accurate in quantity and quality for the reconstruction purposes which must be presentable. The Judges' bias on the digital forensic technology used to present evidence in the court of law in the UK has to be addressed, including the certification processes by the forensic science regulator unit in the UK. The impact of application level information in digital crimes and fraud related activities need to be characterized and models developed to allow for evidence validation. Both of these impacts have differing characteristics depending on the quality of information recovered from the application.

Chapter 3 Research Methodology

3.1. Introduction

This chapter describes the methodology developed for this research project. The possible processes to gain information will be described and the scenarios that have been developed for this research project will be outlined. The techniques for appraising the quantity and quality of information that can be recovered from the application memory will also be detailed. Before detailing the methodology, it is worthwhile re-stating the aims of the research to ensure that the methodology will fulfil those aims. The aims of this research project are;

- To uncover information that may have previously been "hidden" to forensic investigators by extracting application level information from the volatile memory of Windows systems.
- To formulise how the extracted application level information can be reconstructed to describe what user activities had taken place on the application under investigation.

In order to fulfil these aims, certain restrictions have to be made on the project. The first restriction is the operating system that will be used for this investigation. This research project started in August 2008 and at that time, the most dominant operating systems were Windows XP and Windows Vista. Taking a step forward to the present (2011), there are three popular Windows platforms that hold the top three positions in terms of market share; Windows XP (Microsoft, 2007) Windows Vista (Microsoft, 2008) and Windows 7 (Microsoft, 2011). Although Windows XP is the oldest of the three most popular current operating systems, it is still dominant in the market as at June 2011 holding the top position with 55.09% of the market share worldwide; Windows 7 has 23.08%.

However, it is expected that the research methodology used and the statistics that have been acquired can be usefully applied to any applications running on any operating system. As this research project focuses on the information that can be acquired from applications, the most commonly used applications on Windows XP computer systems should be identified. These were identified by contacting three different businesses to ask which applications were most

commonly used on their systems. This question was emailed to the Canadian Imperial Bank of Commerce, Ericsson and the East Thames Housing Corporation. These institutions were selected to represent different business types and because contacts had already been acquired within the business.

The CIBC-Canadian Imperial Bank of Commerce responded that they use Windows XP Service Pack2 (SP2). The commonly used applications are Microsoft Word 2007 (SP2), Microsoft Excel 2007 (SP2), Adobe Reader 8.0, Microsoft PowerPoint 2007 (SP2), Microsoft Outlook 2007, Microsoft Publisher 2007 (SP2), Microsoft Access 2007 (SP2) and, Internet Explorer 7.0. It was mentioned by the Technical Manager that Microsoft Outlook 2007 has the highest use.

Ericsson indicated that it used two operating systems, Windows 2003 and Windows XP. The Windows XP operating systems being used in Ericsson are made up of SP2 and SP3. The most commonly used applications are Microsoft Office 2003 [including Word (SP3), Excel (SP3), PowerPoint (SP2), Access (SP2), Infopath (SP2), Publisher (SP2), Outlook (SP3)]. Other applications are Microsoft Internet Explorer 7.0, Microsoft Dynamics CRM 3.0, Microsoft Office Visio 2003 (SP3), Adobe Acrobat Reader 8, IBM Lotus Sametime 8.0.1 and Steelray Project Viewer 3.8.1.0.

The East Thames Housing Corporation uses Windows XP with Service Pack 2 (SP2). For day to day business operations, the company's commonly used applications are MS Word 2007, MS Excel 2007, MS PowerPoint 2007, MS Outlook 2003, MS Internet Explorer 6, Adobe Acrobat 8 and Sophos Antivirus.

The applications that are the same for all three businesses are MS Word, MS Excel, MS PowerPoint, MS Access, MS Outlook, MS Internet Explorer and Adobe Reader 8.0, although the version numbers differ for each application. When this research project was started, MS Office 2007 was the most deployed office suite in businesses and, as of 2011, it is still the most deployed office suite of applications (Microsoft, 2007). Therefore MS Word 2007, MS Outlook 2007, MS Excel 2007, MS PowerPoint 2007 and MS Access 2007 will be the applications used in this research investigation. In addition to these applications, MS Internet Explorer version 7.0 and Adobe Acroread 8.0 will also be investigated as a part of this research project.

3.2. Volatile memory acquisition

As detailed in the literature review, there are many different tools that can be used to acquire volatile memory. For this research project, it is important to acquire an accurate copy of the volatile memory as easily and as quickly as possible. For this reason, a software based volatile memory acquisition tool was used, and out of those available, Nigilant32 (Matthew, 2008), was selected because it is a freely available tool, it is easy to use and it has a small footprint using less than 1 MB in memory when loaded. As it is a software-based volatile memory acquisition tool, it must be loaded into memory in order to function. This means that there will be an impact by the acquisition tool on the evidence that is acquired. In order to minimise this impact the acquisition tool will be loaded into memory as soon as the computer starts for this research investigation.

3.3. Extraction of allocated memory

After the acquisition of the volatile memory of the Windows system, the memory that has been allocated to the applications under investigation must be extracted. In order to achieve this, the Volatility framework (AAron, 2008) was used. Volatility can identify the processes that were active when the volatile memory was captured. It can also scan through the memory dump looking for closed applications, i.e. applications that had been open, but were closed at the time of acquiring the volatile memory. In addition, this tool is capable of extracting the memory allocated to those processes (closed or open). This means that the process id can be identified for all of the processes associated with each application running on the system at the time of the volatile memory acquisition. Once the process id has been identified the memory allocated to that process can be extracted. Volatility provides access to open and closed applications by means of scanning through the EPROCESS block using the pslist (print list of running processes) and psscan (scan for EPROCESS objects). In this research project, Volatility was used to extract the memory allocated to open and closed processes.

3.4. Extraction of user input

The memory that has been allocated to the application contains all of the data required to run the application and all of the information input by the user. As identified in the literature review, it is not yet known how much user input can be found and how that user input is dispersed throughout the memory. Some method must be used to extract the information related to the user interactions from the volatile memory allocated to the application. The user interaction may be the various mouse clicks that the user makes; it may include video information, sound information, highlights, file information or user text input. Based on the applications that were identified, which mostly required keyboard input, it was decided that user text input, file information and highlights should be investigated for this research project.

In this research project, the first task is to identify the possible format of the keyboard information that has been input by users to each of the applications. There are two types of Unicode formats, UTF-8 and UTF-16. UTF-8 only uses one byte (8 Bits) to encode characters and UTF-16 uses 2 bytes. According to document written by Microsoft Corporation (Microsoft, 2008)**,** Windows applications should use UTF-16 for their internal data representation, therefore, any text representation on a Windows system is expected to use UTF-16 and this is the encoding format that was looked for in the volatile memory allocated to MS Word, MS Excel, MS PowerPoint, MS Outlook Email, MS Access and MS Internet Explorer 7.0.

However, other applications like Adobe Reader 8.0 uses UTF-8 (Microsoft, 2007). An example of what is assumed to be user keyboard input is shown in Figure 3.1. Here the memory that was allocated to the MS Word 2007 application has been opened with "WinHex". This gives an investigator the ability to see the raw information that has been retrieved from memory, but it is odd to have this information seemingly encoded as both UTF-8 and UTF-16. This Unicode information is difficult to find, interpret and analyze using "WinHex" because it is highly dispersed; therefore some other analysis techniques need to be identified. It is possible to extract numbers and Latin characters that are encoded as ASCII, UTF-8 and UTF-16 using an application called "Strings" written by Mark Russinovich and David Solomon (Russinovich & Solomon, 2009). (as ASCII, UTF-8 and UTF-16 all represent numbers and Latin characters by the same value). The "Strings" application was used to convert the memory dump images of

these applications into human readable strings. Extracting only the hexadecimal bytes within the numbers and Latin characters range of ASCII/UTF-8/UTF-16 reduces the size of the file and thereby making it easier to work with during analysis. However, it was found that the files generated by using the "Strings" application did not contain evidence that could be found in the original acquired volatile memory image.

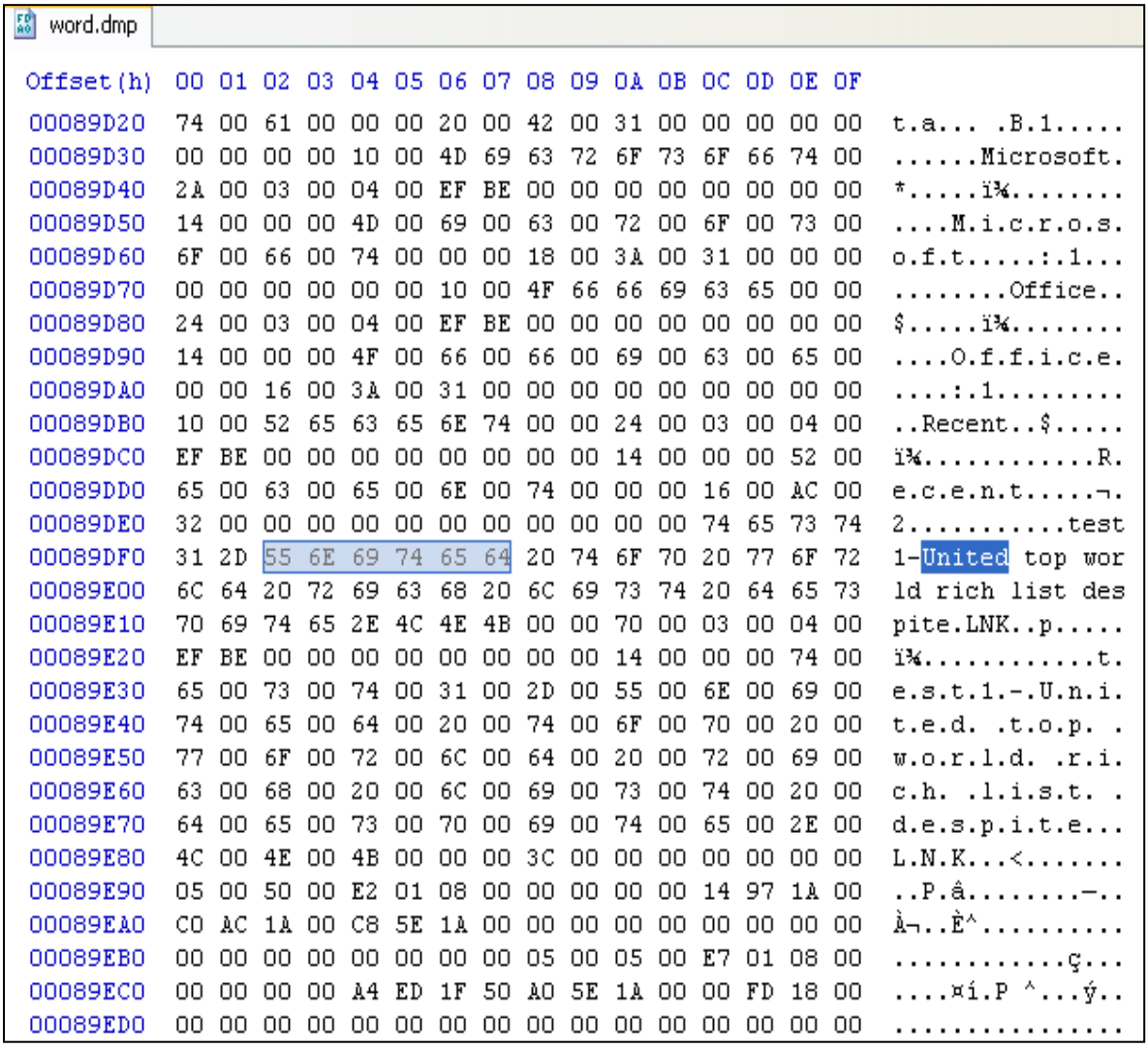


Figure: 3.1 WinHex of the memory dump for Word 2007

The application "Strings" only identifies sequences of data as a string if 3 or more consecutive characters are seen as strings values. The data that was missing was identified in the original acquired volatile memory image as being more than 3 consecutive characters so it is unknown

why this occurred and it is viewed that strings is not a forensically sound application. It was therefore decided to write a program that would extract string information from the volatile memory that had been allocated to the process. In order to achieve this, a simple conversion program was written which would extract only those hexadecimal bytes (ASCII, 2007) of information within the range 32_{10} to 126_{10} inclusive (i.e. only numbers, Latin characters and punctuation, not any control characters or characters from other languages). It should be noted that the values extracted within this range may not be text information (it may be program code for instance), but it was assumed that the user input may be extracted using this method. After the string information has been extracted it is processed in two ways.

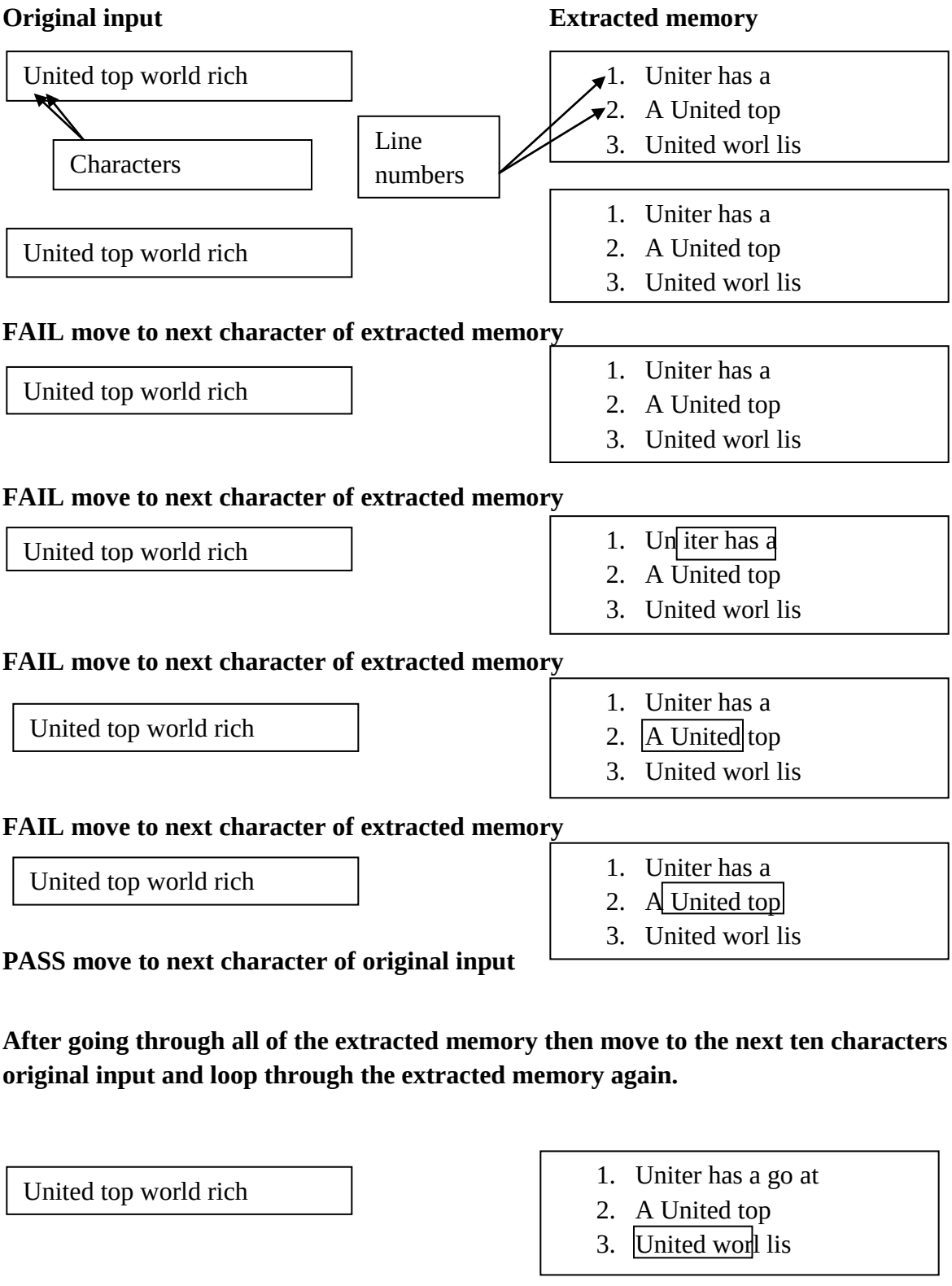
In the first way, the extracted application level information was matched to the information that was known to occur in memory. In this case, the original user input had been recorded and this was used to find the same information in memory. This was done to attempt to identify how the original user input was dispersed in the memory and how easy it is to identify. This will aid future investigations by providing an indication of the probability of finding relevant information associated to the user input.

The second way that the information was processed was to search for commonly used English words. The aim was to identify the same amount of information using this technique as when the user input is unknown. These commonly used English words could, of course, be replaced by specific words related to an investigation.

3.4.1. Pattern matching

Pattern matching is the act of finding a subsequence of symbols within a sequence of symbols. This implies that the various user inputs that were made on each of the applications were recorded before images were captured.

According to (NIST, 2009) guidelines on text searching, the searches are performed by comparing the search criteria specified by the user against the information stored in the memory of the application and for the type of search specified as detailed below.



After going through all of the extracted memory then move to the next ten characters of the original input and loop through the extracted memory again.

The originally recorded user input was matched and was first investigated and then commonly used English words.

3.4.2. Commonly used English words

The idea of commonly used English words came from the understanding of the meaning of autonomy's based computing infrastructure technology, the intelligent data operating layer (IDOL) of computer system (Autonomy, 2008). This layer can form a conceptual and contextual understanding of all content of data in a system and indexing content formats.

This intelligent data operating layer (IDOL) can be used to automatically analyze any piece of information from different data sets. The use of an autonomy layer can automatically produce a brief summary of each piece of content that is returned for a query. It generates three different types of summaries:

1. The conceptual summary displays a few sentences from the document that contain the most salient concepts (these sentences can be from different parts of the result document).
2. The contextual summary relates to the context of the original query allowing the most applicable, dynamic summary to be provided in the results of a given query.
3. The simple summary comprises a few sentences of the result document.

The understanding of the use of the intelligent data operating layer (IDOL) can enable search across the entire computer system.

Additionally, the idea for the use of some commonly used English words to search for the user input stored on Windows applications came from the Oxford English Corpus. According to (Oxford, 2011), which focused on the most frequently used English words from the Oxford English Corpus, and also presented by (Peter, 2011), the top 100 commonly used English words are *"the, be, to, of, and, a, in, that, have, I, it, for, not, on, with, he, as, you, do, at, this, but, his, by, from, they, we, say, her, she, or, an, will, my, one, all, would, there, their, what, so, up, out, if, about, who, get, which, go, me, when, make, can, like, time, no, just, him, know, take, people, into, year, your, good, some, could, them, see, other, than, then, now, look, only, come, its, over, think, also, back, after, use, two, how, our, work, first, well, way, even, new, want, because, any, these, give, day, most, us"*. Out of all these 100 most commonly used English words, ten were used to represent commonly used English words that were searched for in the extracted volatile memory images of sample applications in this research project. The ten commonly used English words are: *and, in, is, as, the, on, of, to, be, that.*

3.5. Experiment details

In order to make the results as applicable as possible, the experiments were designed to try to replicate a normal working environment. The computer system running Windows XP would be turned on at the start of the day and then turned off at the end of the day. Before the experiments started, the Windows machine will be shut down and rebooted to ensure that the system was as "clean" as possible. This is also necessary to ensure that the memory allocated to each application had not previously been used to store unrelated data. Throughout the experiment all the applications will be running or closed while collecting data.

To ensure that the allocated memory can be easily recovered, the process identity of each capture will be recorded. Although the amount of knowledge an investigator has before conducting this investigation may be varied, it is unlikely to include knowledge of all of the evidence the investigators are looking for. Fragments of user information (for example, documents they were working on and the web pages they viewed), as stored in various areas of memory of the applications, will be fully explored.

The user input was based on information found in newspapers on the day that the experiment was conducted. Four scenarios have been designed so that the results will help investigators when they are faced with collecting data from a computer in different situations.

1. Scenario 1: The investigator finds the computer still turned on, the applications that the user was using are still open and the user has recently interacted with the applications. For this scenario, the applications are opened at the beginning of the day, the user uses the applications as if they were working on a normal day and the volatile memory images are captured at set interval of 30 minutes.

2. Scenario 2: The investigator finds the computer still turned on, the applications that user was using are still open and the user has not recently interacted with the applications. For this scenario, the applications are opened at the beginning of the day; the user uses the application at the start of the day and then does not interact with the applications. The volatile memory images are captured every 30 minutes.

3. Scenario 3: The investigator finds the computer still turned on, but the applications that user was using are now closed. However, the computer has not been used for any other purpose. In this scenario, the user interacts with the application at zero minutes and then the applications are closed, no other user input is allowed, i.e. there is no user interaction with the system. The volatile memory was captured every 30 minutes.

4. Scenario 4: The investigator finds the computer still turned on, but the applications that user was using are now closed. However, the computer has been used consistently during the day for other purposes. In this scenario, the user interacts with the application at zero minutes and then the applications are closed. There is further user interaction with the system, i.e. user uses the system to run other applications. The volatile memory was captured at every 30 minutes.

The research investigation will focus on what user input applications may contain and how easy it is to find and reconstruct this information.

3.5.1. Scenario 1

This investigation focused on answering the question "can all information related to how a user is using that application be recovered if the memory is captured while that application is still running?".

The ability to identify information that can be recovered from the memory of an application will add value to a forensic investigation. For example, a research work of (Eoghan & Richard, 2010) focused on extracting Windows command line details from physical memory. This technique can be used as a prime source of evidence in many intrusions and computer crimes to revealing important details about the offender's activities on the subject prompt.

However, there is a limitation to the amount of relevant information recovered from the command history. While this investigation doesn't look at command line history, the scenario was designed to look at user information that can be recovered from the application and the information found will be expected to be just as useful.

In Scenario 1, the user input on each of the applications is different at a set interval of 30 minutes and volatile memory was captured every 30 minutes while the applications are still running. Table 3.1 describes typical user input for scenario 1.

Table 3.1 Sample user input for scenario 1

Application	**User input action(s)**
Word 2007	Write a paragraph of text or do nothing on the document. Save the document or do not save.
Excel 2007	List a set of numbers, draw a graph of the numbers or do nothing. Save the document or do not save. Input may contain alphanumeric characters.
Outlook 2007	Send and receive emails or do nothing. Save the email or do not save.
PowerPoint 2007	Write a slide or slides of text or do nothing. Save the document or do not save.
MS Access	Write text and numbers into a database or do nothing. Save the database or do not save
Adobe Reader 8.0	Highlight text, search for texts or do nothing. Save the document or do not save.
IE 7.0	Open websites and follow hyperlinks. Click backwards or forwards in the web browser. Save the webpage or do not save. Highlights texts, search for texts or do nothing.

3.5.2. Scenario 2

This investigation focused on how much data is lost if the application is running, but the user is not interacting with the application. This second scenario was designed to investigate the time aspect of information stored by an application and how much data is retained that can be recovered if the application is still running. The approach taken for user input on Scenario 2 is different to Scenario 1. For example, in Scenario 2, user input was made once by a user and no other inputs were made, but the applications were still running. Volatile memory images were captured at set interval of 30 minutes. Table 3.2 shows the typical user input for Scenario 2.

Table 3.2 Sample user input for scenario 2

Application	**User input action(s)**
Word 2007	Write a paragraph of text on the document.
Excel 2007	List a set of numbers and draw a graph of the numbers. Input contains alphanumeric characters.
Outlook 2007	Send and receive emails.
PowerPoint 2007	Write slides of text.
MS Access	Write text and numbers into a database.
Adobe Reader 8.0	Highlight text and search for texts.
IE 7.0	Open websites and follow hyperlinks. Click backwards and forwards in the web browser.

3.5.3. Scenario 3

This investigation focused on how much data is lost if the application is closed and the system is not used for anything else. In a related research work, (Andreas, 2006.), a search process to find active system objects was designed. This approach was proven successful in identifying system objects, files and connections on closed applications of Windows. In addition, a research project of (Jason, Ewa, Derek, & Magdalena, 2007) focused on user data persistence in physical memory and the time that the allocated memory pages can be retained in the memory. This included the ability to measure the time-stamps of data segments and block device cache pages that are persistent in the memory for less than 5 minutes, which makes it impossible to determine the age of text segment pages.

Therefore, Scenario 3 was designed to identify the time aspect of user input stored by an application and how much information can be recovered from the memory when applications are closed. The approach taken for user input on this Scenario 3 is different to Scenario 1 because, user input was made once by the user and no other inputs were made. Volatile memory images were captured at a set interval of 30minutes when applications have been closed. Table 3.3 describes typical user input on application that was designed for Scenario 3 in this research project.

Table 3.3 Sample user input for scenario 3

Application	**User input action(s)**
Word 2007	Write a paragraph of text on the document.
Excel 2007	List a set of numbers and draw a graph of the numbers. Input contains alphanumeric characters.
Outlook 2007	Send and receive emails.
PowerPoint 2007	Write slides of text.
MS Access	Write text and numbers into a database.
Adobe Reader 8.0	Highlight text and search for texts.
IE 7.0	Open websites and follow hyperlinks. Click backwards and forwards in the web browser.

3.5.4. Scenario 4

This investigation focused on how much data is lost if the application is closed and the system is used to run other applications. In this scenario, the investigation was designed to uncover useful information such as the fact that a certain amount of memory allocated to user input may be recovered even after the Windows applications have been closed and the user continued to interact with the system.

The approach taken for user input on the other scenarios are different in scenario 4, user input was made once by a user and no other inputs were made. However, while the applications were closed, the system was used to run other applications and volatile memory was captured at set intervals of 30 minutes.

Table 3.4 describes typical user input on the application for scenario 4. The system is used to run ten different other applications when the seven most commonly used applications have been closed.

Table 3.4 Sample User input for scenario 4

Application	**User input action(s)**
Word 2007	Write a paragraph of text on the document.
Excel 2007	List a set of numbers and draw a graph of the numbers. Input contains alphanumeric characters.
Outlook 2007	Send and receive emails.
PowerPoint 2007	Write slides of text.
MS Access	Write text and numbers into a database.
Adobe Reader 8.0	Highlight text and search for texts.
IE 7.0	Open websites and follow hyperlinks. Click backwards and forwards in the web browser.

3.6. Size of memory extracted

As described in Section 3.5, four different scenarios were investigated; during these scenarios the volatile memory allocated to the seven most commonly used applications was captured. The number of memory captures for each of the four scenarios was fixed, 100 captures were made for each scenario. Figure 3.5 details the average size of the memory extracted from Windows applications for each of the four scenarios.

Table 3.5 The size of the memory extracted from application level information

Applications	Number of volatile memory captures	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
		Size (GB)	Size (GB)	Size (GB)	Size (GB)
MS Word	100	477	467	440	811
MS Excel	100	171	582	410	391
MS PowerPoint	100	295	313	313	196
MS Outlook Email	100	419	641	235	414
MS Access	100	225	332	401	366
MS Internet Explorer 7.0	100	464	431	337	303
Adobe Reader 8.0	100	355	628	747	641

3.7. Quantitative measures

A quantitative assessment of the information that is recoverable from the captured memory could indicate the ease with which information can be found.

The quantitative assessment measures that were formulized for the purpose of this research project are:

1. Mean length of the initial user input
2. Mean number of times user input was repeated in the memory
3. Percentage of user input was found in the memory
4. Mean length of user input was found in continuous blocks

These quantitative techniques are useful in the evaluation of the user input extracted from the volatile memory of an application.

3.7.1. Mean length of the initial user input

The mean length of the initial user input is the length of the user input on an application. User input may contain alphanumeric text input and the number of the characters are counted for each capture, giving an overall mean value of the length of input made by the user on each application.

For example:

Original user input in Test1 = “The green jacket”. Character = 16
Original user input in Test2 = “The blue jacket”. Character = 15
Original user input in Test3 = “ ”. These NULL inputs are ignored

The metric is the mean number of characters in the original user inputs. This metric is used to find a correlation between the length of user input and the possibility of finding that input in volatile memory.

3.7.2. Mean number of times a character is repeated

User input is repeated in the memory allocated to the application. This metric is the mean number of repeated characters of user input stored.

The metric gives an indication of the number of fragments that have been found in memory by matching the characters of the original user input to the fragments found in memory.

For example:

Original user input Test1 = "The book contains sometimes".
What is found in the memory = "The book"
"The book contains cont"
"Somet somet"

The number of times the first character "o" in "book" is repeated is 2.

3.7.3. Percentage of user input found in the memory

To calculate the percentage of user input found in the memory, the amount of the original user input found in the memory is divided by the number of characters of the original input.

For example:

Original user input Test1 = "The book contains sometimes".
What is found in the memory = "The book"
"The book contains cont"
"Somet somet"

In this case the three fragments of information found in memory do not include all of the information that was originally input. The original user input was 27 characters and only 23 characters were found in memory so, 85% of the original user input has been found.

3.7.4. Mean length of user input found in continuous blocks

The mean length of user input found in continuous blocks is the mean length of the user input which is found as stored in contiguous bytes in the volatile memory.

For example:

Original user input Test1 = "The Old Trafford side, who are more than".

What is found in the memory = "The Old Traff".

"rafford side, who are".

"de, who are mor".

In this case, the three fragments of information found in memory are 13, 21 and 15 characters as is size. The mean length of user input found in continuous blocks is 16. This metric can provide an indication of the ease with which the original user input can be found.

3.8. Qualitative assessment

The quality of the extracted application level information will be reconstructed and by reconstructing the user input activities on applications the forensic investigators will be able to gain access to further information about the user.

The method of searching for fragments of information when user input is known will be compared and contrasted with the novel approach of searching through memory using the most commonly used English words; i.e. when user input is unknown.

3.8.1. Known user input

Figure 3.2 illustrates a sample result of known information in the search for user input on volatile memory of an application. As shown in this experiment, there are lots of duplicates and missing information related to user input on the application.

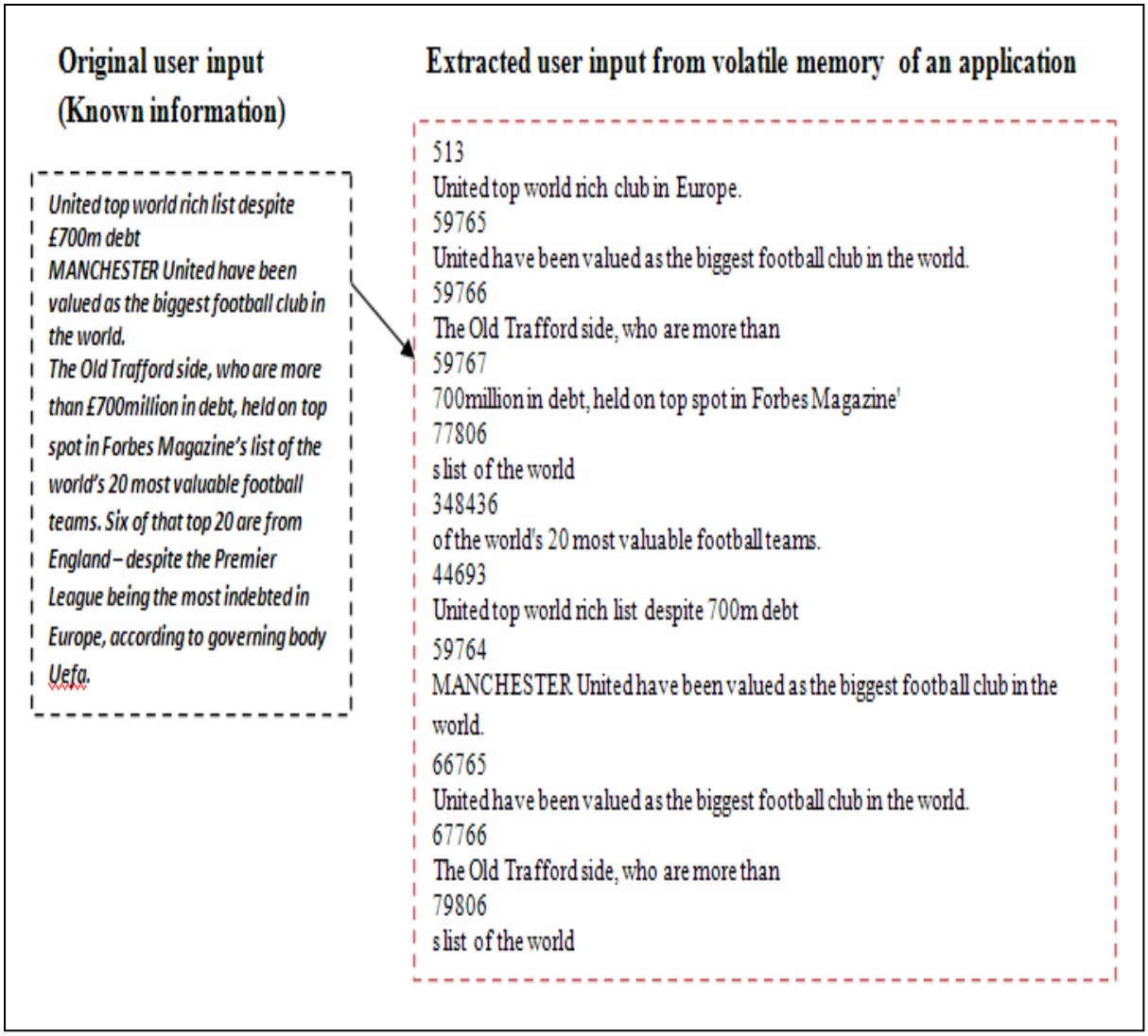


Figure 3.2 Example of search result of known information of user input

3.8.2. Unknown user input

The qualitative measure of unknown user input includes the use of commonly used English words**.** The use of ten commonly used English words have been adopted (Oxford, 2011), (Peter, 2011) and these words, could, of course, be replaced by some words related to the investigation. In this experiment, detailed information about user input is found in the allocated memory of the application including some additional related useful information to forensic investigators. Figure 3.3 illustrates a sample result of unknown information method of searching for user input in the volatile memory of an application.

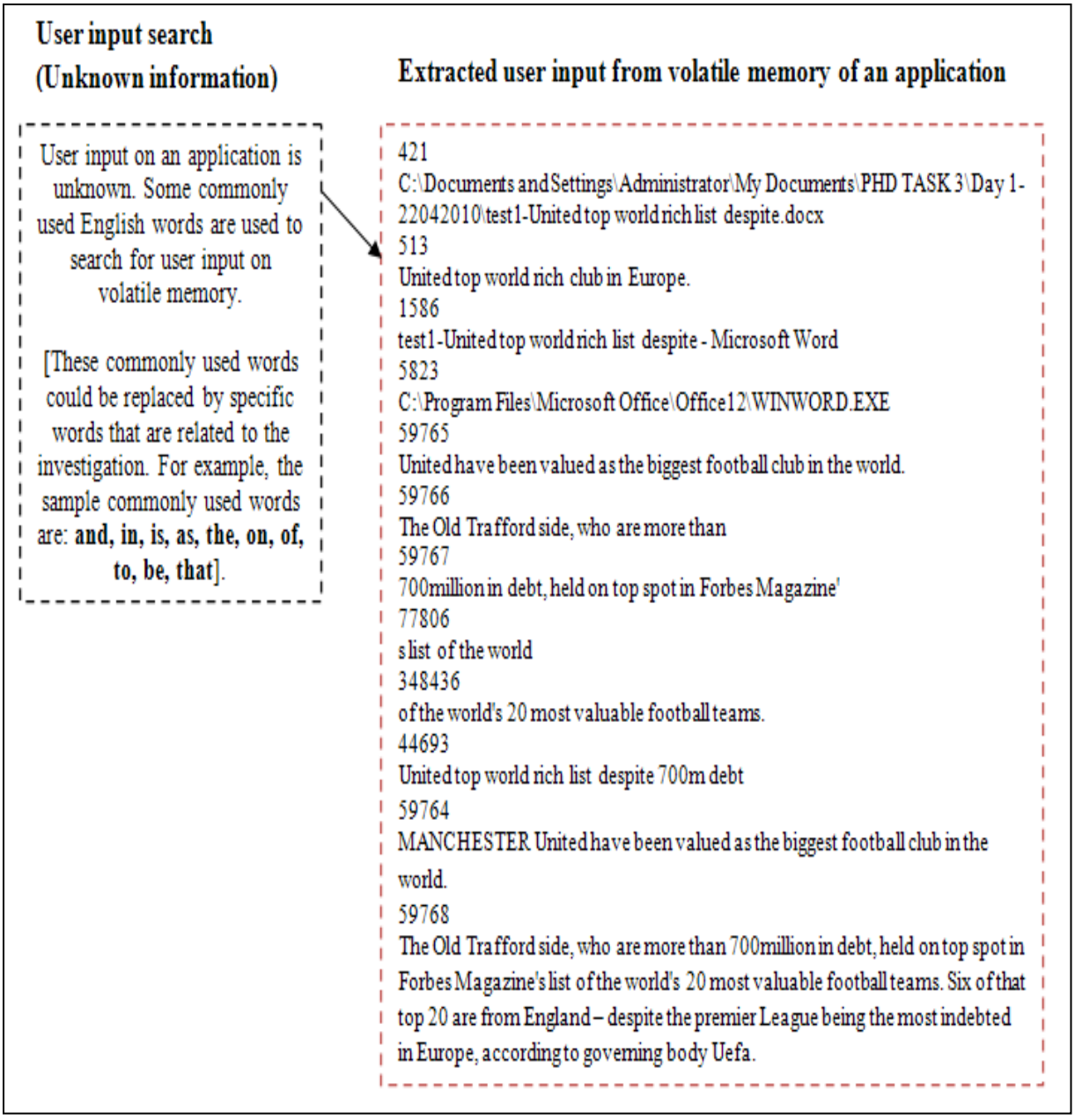


Figure 3.3 Example of search result of unknown information of user input

3.8.3. Reconstruction of user activities

The reconstruction of user input involves sifting through the extracted application level information in order to determine what events occurred. These are presented in two different ways. First, the sample result of known information that contains partial fragment of user input on the applications as illustrated in Figure 3.4.

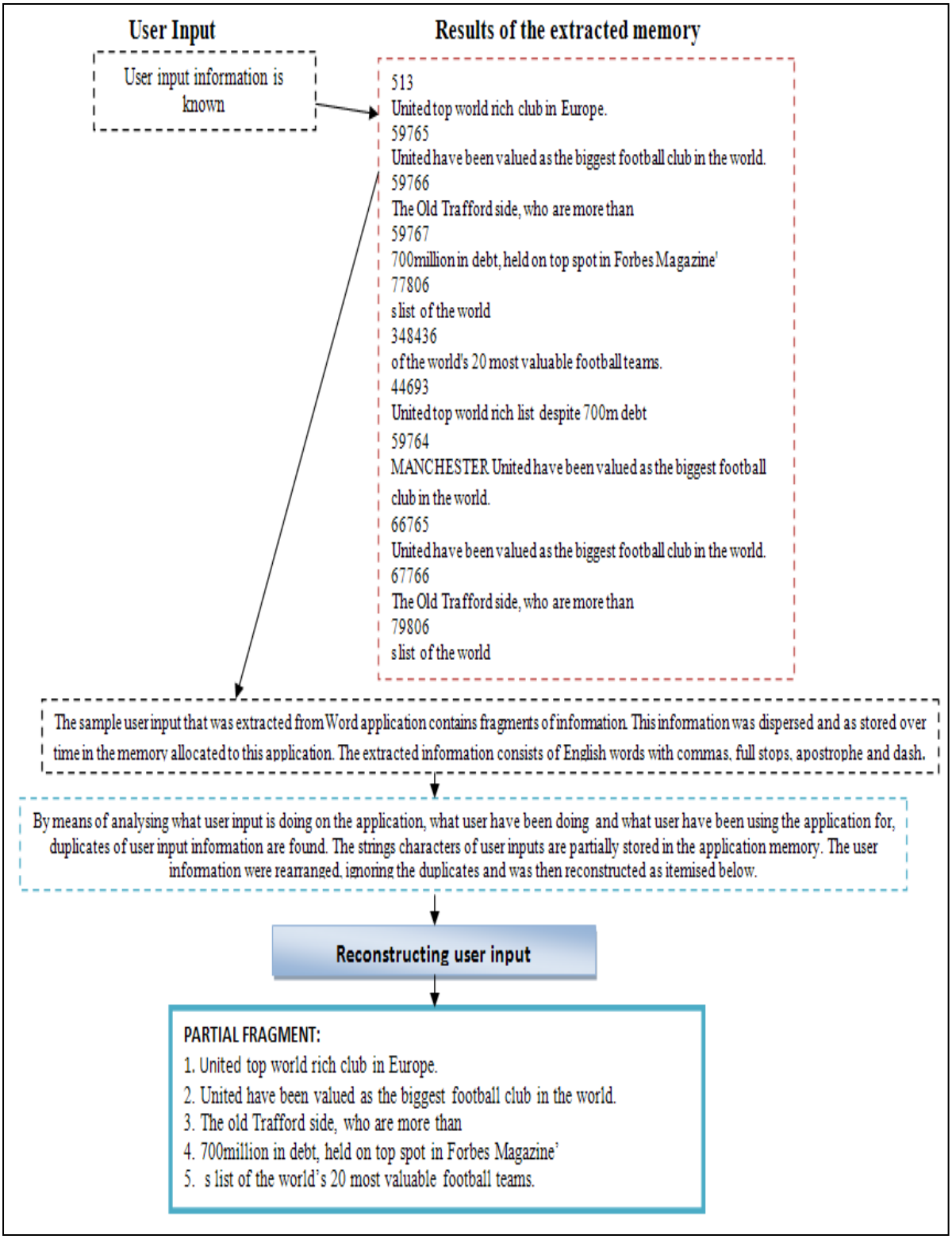


Figure 3.4 Sample reconstruction of user activities from pattern matching with known information

Second, the sample result of using unknown information to search for the user input from the allocated memory are presented in Figure 3.5. This contains whole fragment of user input including some additional information that can be useful to forensic investigators.

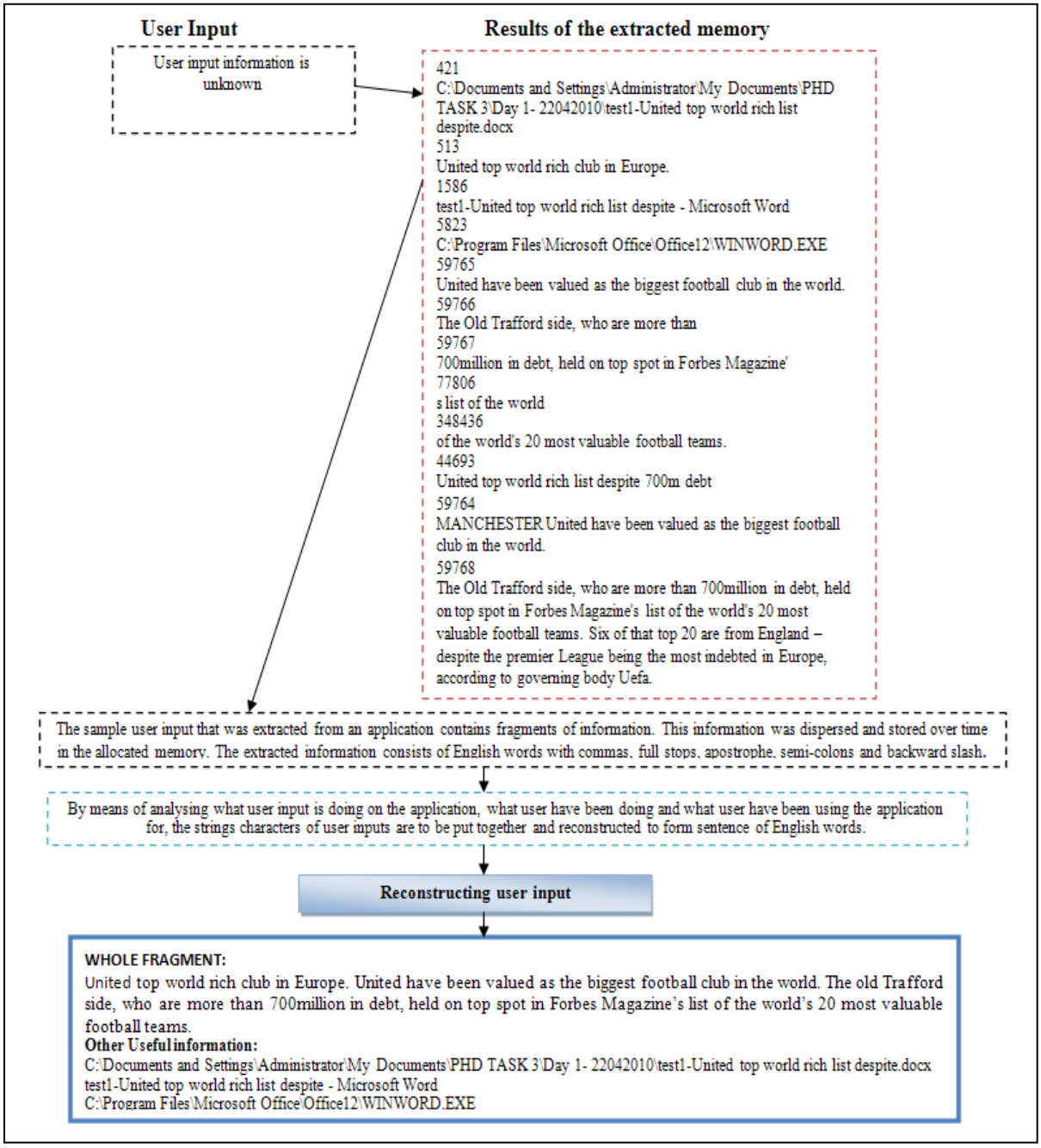


Figure 3.5 Sample reconstructions of user activities of unknown information

The sample reconstruction of user input activities might be arranged and put together in a different order. In this reconstruction of user activities, the extracted application level information was rearranged and put together as recovered from the memory allocated to the application. For example, the whole fragment of information was put together to form a words of sentence or sentences, while partial fragments will be put together in an order of findings.

To maintain the consistency of data for the case at hand the user input recovered was copied into Notepad. This information was examined and re-examined over a period of time. For reconstruction purposes by physically sifting through the user input recovered on each of the applications. This process continues on Notepad as the arrangement of user input recovered involves joining together sentences then arranging and re-arranging the related user input. This process of arranging and rearranging the user input was done for a series of days on each of the applications until whole fragment of related data were put together to form the basis of searching for data based on what user is doing on the application. This information was termed as application level information found on application memory.

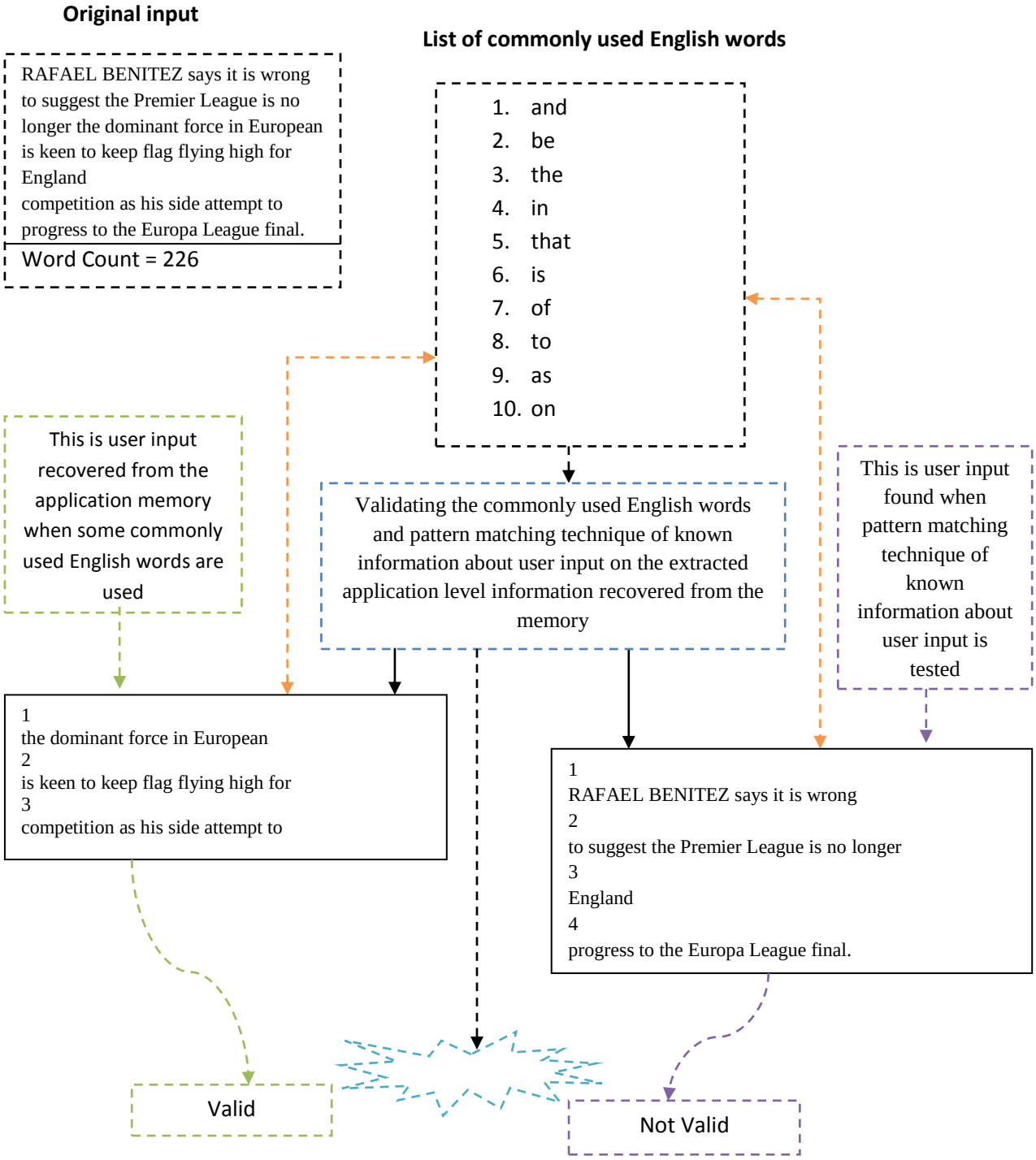
3.8.4. Validation of the quantitative metrics\qualitative assessments

In this research project, the quantitative metrics were designed as stated above for the purpose of searching for the amount of user input that is stored in the application memory. Also, the qualitative assessments were discussed. The quantitative metrics and qualitative assessments were designed after the reviewing of the guidelines of the National Institute of Standards and Technology (NIST), which are based on the forensic string search specification (NIST, 2009).

In addition, the general guidelines for the (SWGDE) Scientific Working Group on Digital Evidence (SWGDE, 2010) were reviewed for the purpose of validation. Following the reviews of the guidelines, the simple program that was developed has been validated. The sample metrics have been used to quantify the user input that have been calculated and tested. The qualitative assessment has also been used for the reconstruction of user activities on the application. The amount of related user input can be used to infer actions in the allocated memory of an application.

3.8.5. Sample validation of the pattern searching techniques

A simple program that was designed for searching for user input on the application memory was validated and tested using the technique of some commonly used English words and pattern matching technique of when user input is known.



3.9. Summary

This chapter has detailed the methodology that will be taken in this research project. Four scenarios will be investigated and the methods of analyzing the data that is captured have been outlined. The pattern matching technique for finding user input on Windows applications was described. The reconstruction of user input activities and the novel approach of using commonly used English words as search terms were presented.

Chapter 4 Quantitative Results and Analysis

4.1. Introduction

This chapter presents the results of the quantitative assessment of application level information from volatile memory of Windows applications. The four scenarios detailed in Chapter 3 were carried out and the four metrics designed were used in the analysis. The results of each of the scenarios will be presented. The quantitative metrics were computed and recorded as detailed in Chapter 3. The metrics were used to describe the amount of user input recoverable from the applications and the ease with which that information can be recovered. The amount of user input recovered can be used to infer the amount of related user actions in the memory allocated to the applications. The experiments carried out on each of the scenarios are discussed below. An aim was to identify the same amount of information using the technique of searching with commonly used English words as was found when the original user input is known. This has been investigated and presented for each of the scenarios in this research project.

4.2. Scenario 1

In this scenario, the user input was recorded while the applications were opened and actively running on Windows systems. As this occurs, images were captured at set intervals until 100 measurements of images were taken. The investigation focused on a specific question, *"*can all information related to how a user is using the application be recovered if the memory is captured while that application is still running?"

4.2.1. Mean values of each metric

Table 4.1 presents the quantitative results of the seven most commonly used applications identified in this research project. The mean of each of the metrics are presented as found in the investigation.

Table 4.1 Mean values of metrics measured

Types of Application	Volatile Memory Capture	Length of User Input	Number of Times Character of User Input is Repeated	Percentage of User Input found	Length of User Input found in Continuous block
Word	100	477.06	194.00	96%	48.65
Excel	100	171.47	51.65	44%	21.33
PowerPoint	100	295.70	110.00	95%	51.89
Outlook Email	100	419.73	275.22	95%	24.59
MS Access	100	255.05	453.00	39%	17.22
IE70	100	464.83	410.67	97%	37.40
Adobe	100	355.79	215.84	35%	34.48

4.2.2. Adobe Reader 8.0

The experiment was run as defined in the methodology and the three metrics of graphs were plotted as below.

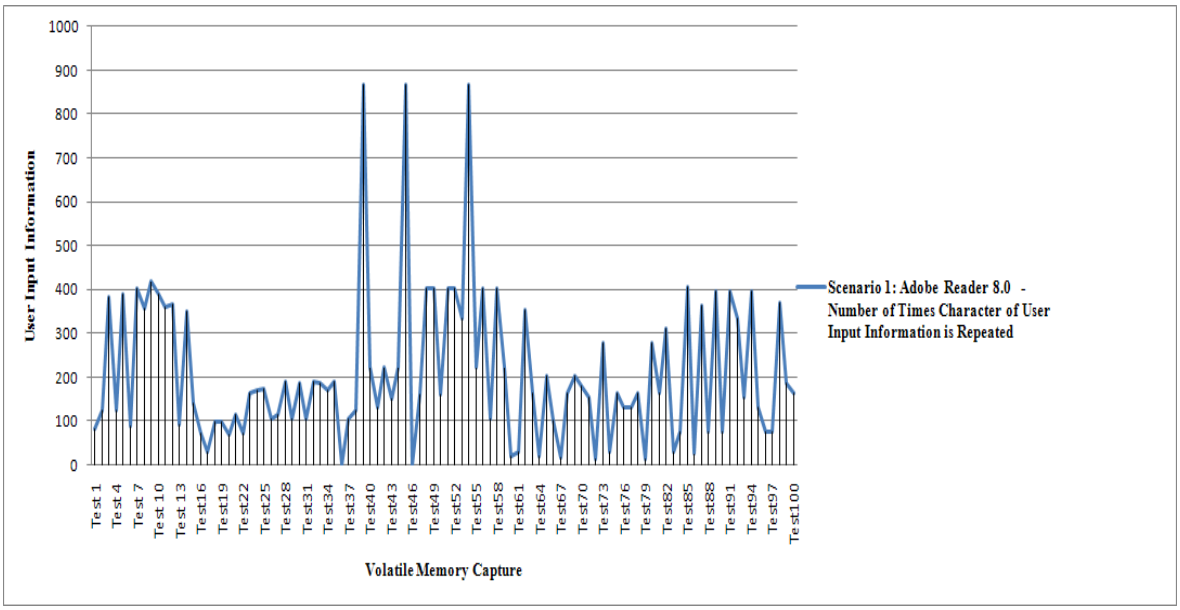


Figure 4.1 Number of times a character of user input is repeated

As shown in Figure 4.1, test39, test45 and test53 recorded the highest number of times a character of user input is repeated whereas, the percentage of user input found for each of these tests were not the highest, in Figure 4.2.

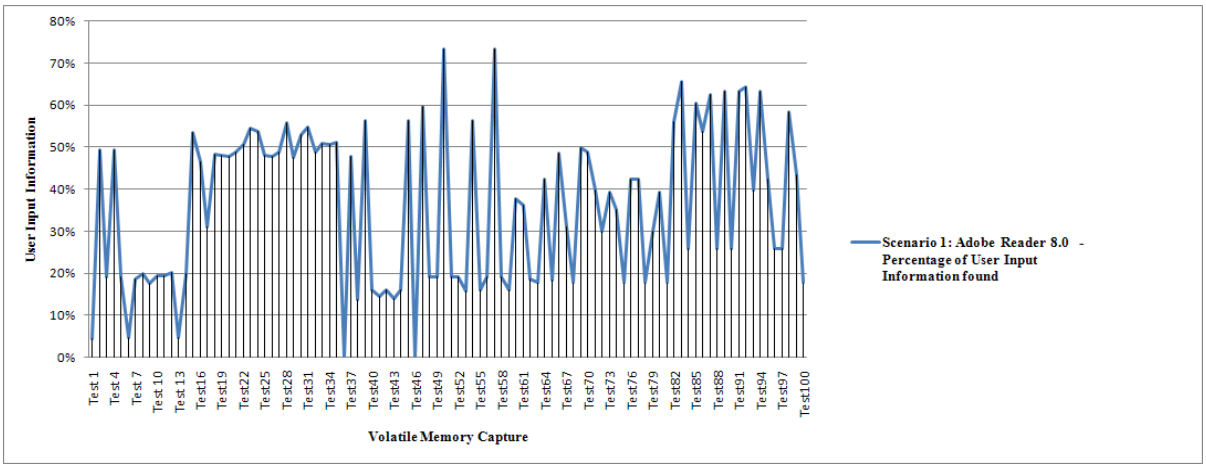


Figure 4.2 Percentage of user input found

The information found in continuous blocks of the memory in Figure 4.3 recorded the highest value in test41, test53 and test59. These values are distinctively larger than other scenarios.

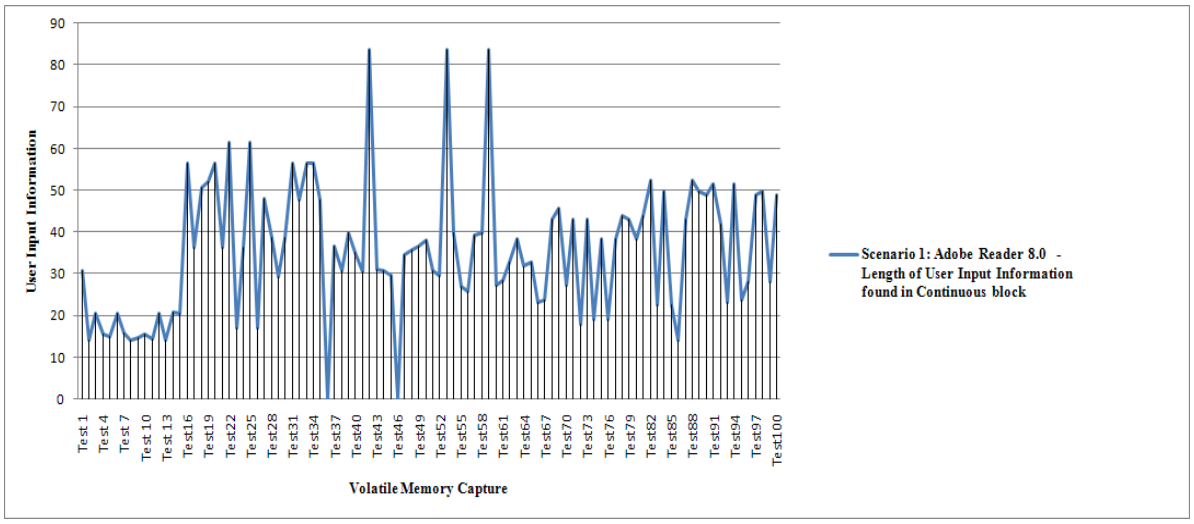


Figure 4.3 Length of user input found in continuous block

However, the percentage of user input found in test41 is relatively low at just above 10% with a similar finding for test53 and test59, where the percentage was approximately 20%. This is thought to be due to the difference in user interaction between the different tests of the

experiments. For example, in test39, test45 and test53 more short pieces of text were highlighted than in test41. There was no correlation found between the overall length of user input and in any of the three metrics presented here. When comparing the two patterns searching techniques, there is 32% of user input found when using the pattern of some commonly used words whereas 35% user input was found when the pattern of known information about original user input was used.

4.2.3. MS Word

In Scenario 1, the extracted application level information of Word application was investigated and the three metrics were plotted, as shown below:

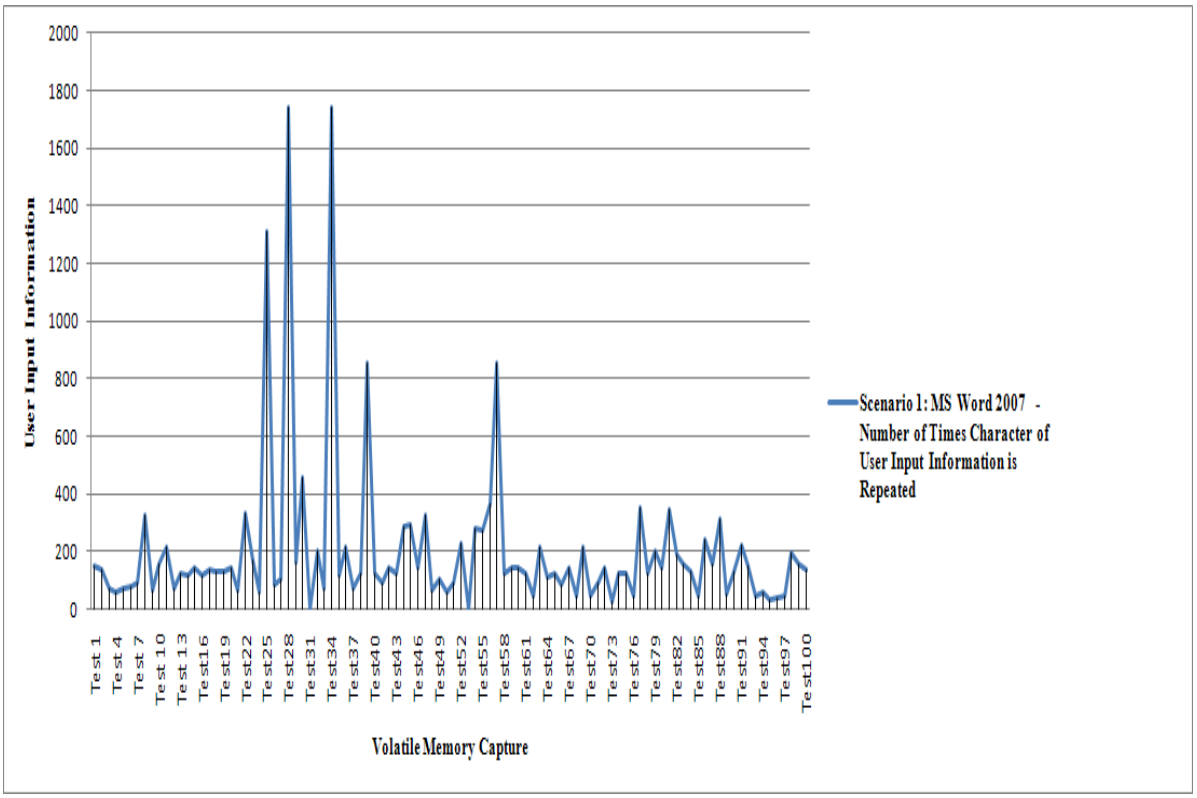


Figure 4.4 Number of times a character of user input is repeated

In Figure 4.4, the number of times a character of user input is repeated is at the highest in test28, test34. The percentage of user input found was between 90% and 100% for all tests where user input was made, as shown in Figure 4.5. It can be said that nearly all related user input can be recovered from the application memory when the user is using MS Word.

However, this statistic does not indicate the quality of the information that has been recovered, i.e. whether or not the information can be reconstructed to identify what the user has been doing, merely that the same character sequences as were input can be found in memory.

In this experiment, the search pattern technique was applied, 90% of user input was found when using the pattern of some commonly used English words to search for user input stored in the volatile memory whereas, when pattern of known information about the original user input was used, 96% of user inputs were recovered.

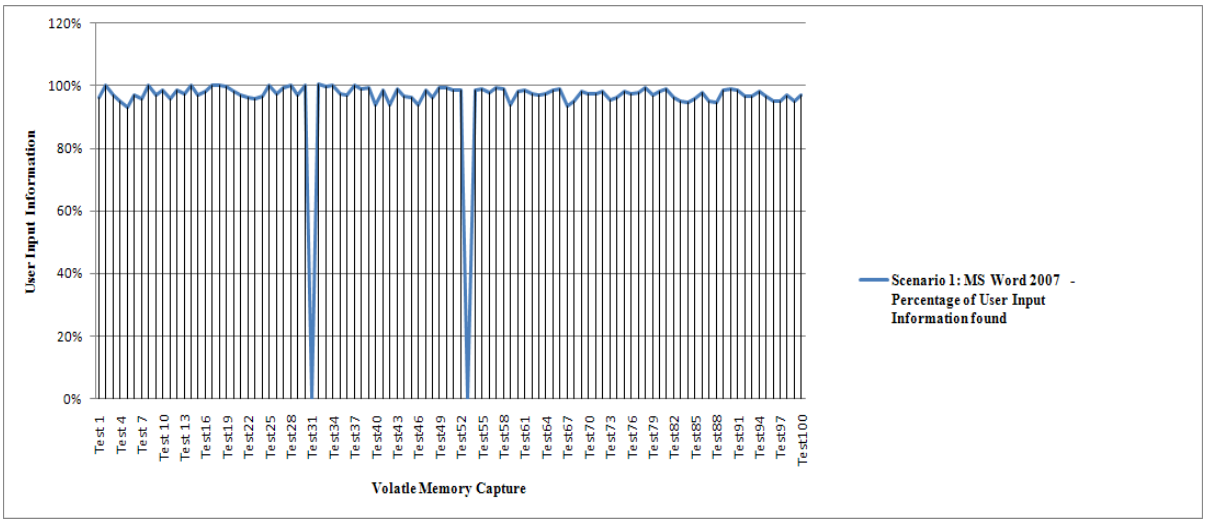


Figure 4.5 Percentage of user input found

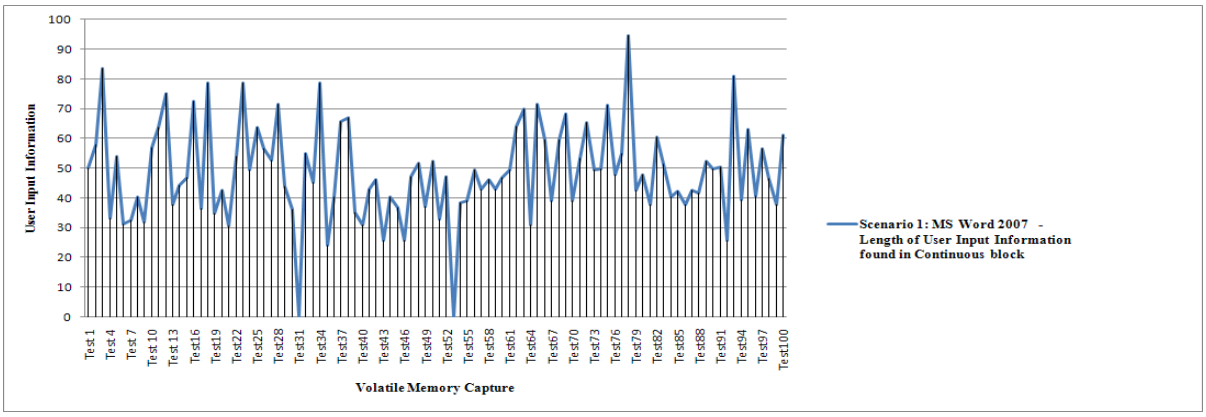


Figure 4.6 Length of user input found in continuous block

In Figure 4.6, the information found in continuous blocks of the memory is almost always larger than 30 characters in length which indicates that it is easy to find information in memory and that the information found might be more useful (if it is formed of contiguous blocks of characters, reconstruction is easier). Again, there was no correlation found between the overall length of user input and in any of the three metrics presented here, but there are more user input found in continuous blocks.

4.2.4. MS Excel

Three different graphs were plotted to illustrate the quantitative assessment of user input stored in the volatile memory of MS Excel application. There is a large amount of in-built system defined data in the application memory (See Appendix B) which can make it difficult to find data when using Excel. In this scenario, user input on Excel contains a greater mixture of Latin characters and numerical data when compared to using the other applications. It proved to be very difficult to identify the numerical data because of the existence of other numerical data in the memory image that was captured. In appendix B, a sample of an Excel spreadsheet that contains other numerical data existing in the memory is presented. It is assumed, therefore, that the variation exhibited in the graphs is due to the relative amount of user input which was Latin characters (easier to find) to numerical characters (difficult to find).

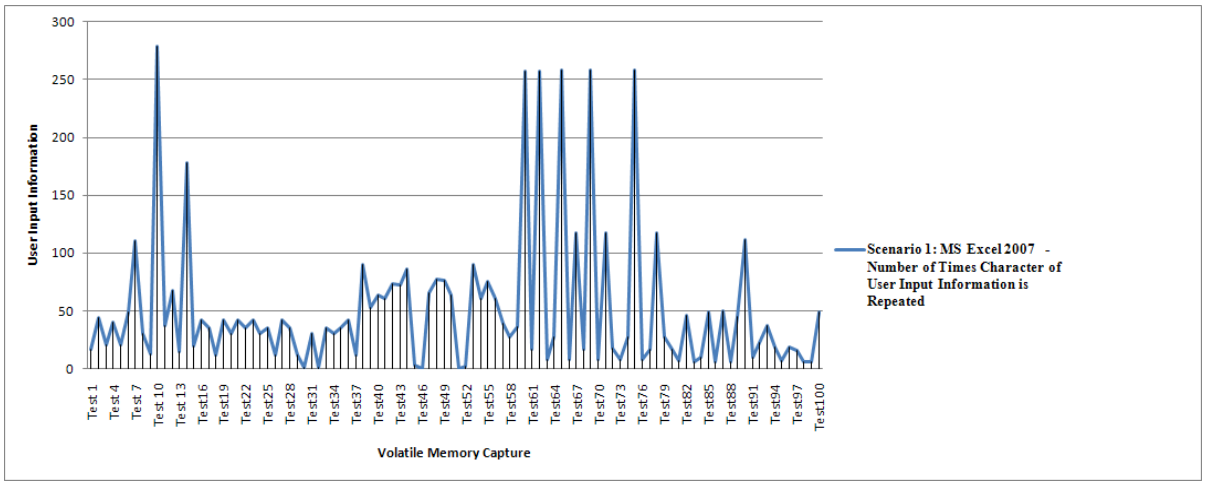


Figure 4.7 Number of times a character of user input is repeated

In Figure 4.7, test10, test59, test62, test65 and test74 reported the highest peak of the number of times a character of user input is repeated.

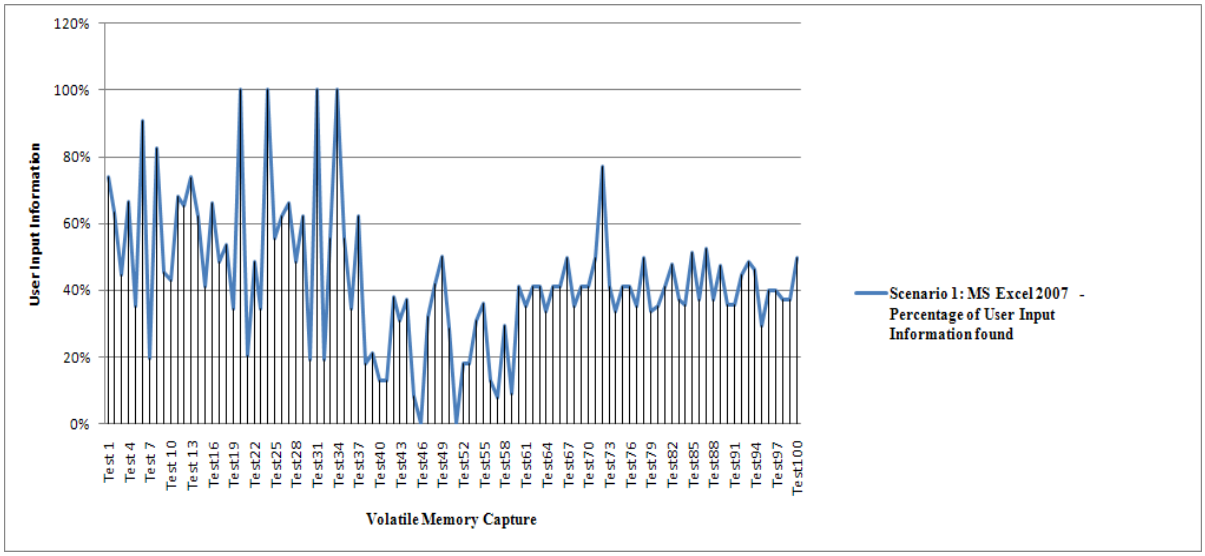


Figure 4.8 Percentage of user input found

In Figure 4.8, test19, test24, test31 and test34 reported the highest peak of the percentage of user input found. It can be said that the user input contain more of textual characters than the numerical characters.

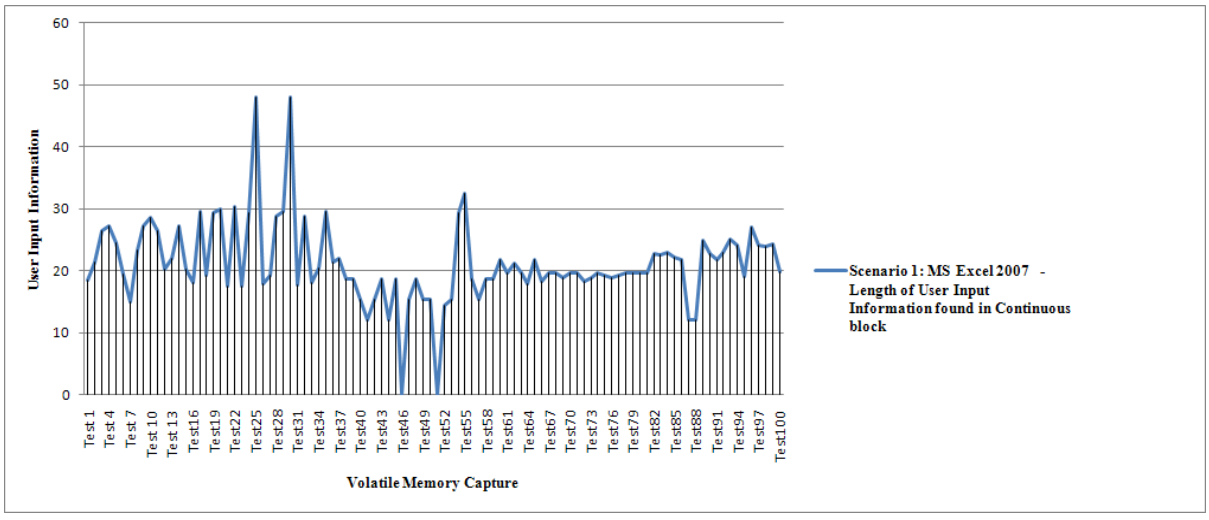


Figure 4.9 Length of user input found in continuous block

In Figure 4.9, similar result was reported in test24 and test31. It was shown that the length of user input found in continuous block is at highest peak. This means more of textual data was initially entered on the application. The search pattern techniques revealed some vital information. When using the pattern searching of some commonly used English words, 40% of user input was recovered whereas, the pattern matching of when user input is known resulted in 44% of user input that was recovered from the memory.

4.2.5. MS Outlook Email

The results of the quantitative assessment are shown below. Figure 4.10 shows the number of times a character of user input is repeated in the memory and test42 and test54 recorded the highest value of this metric. This is followed by test28.

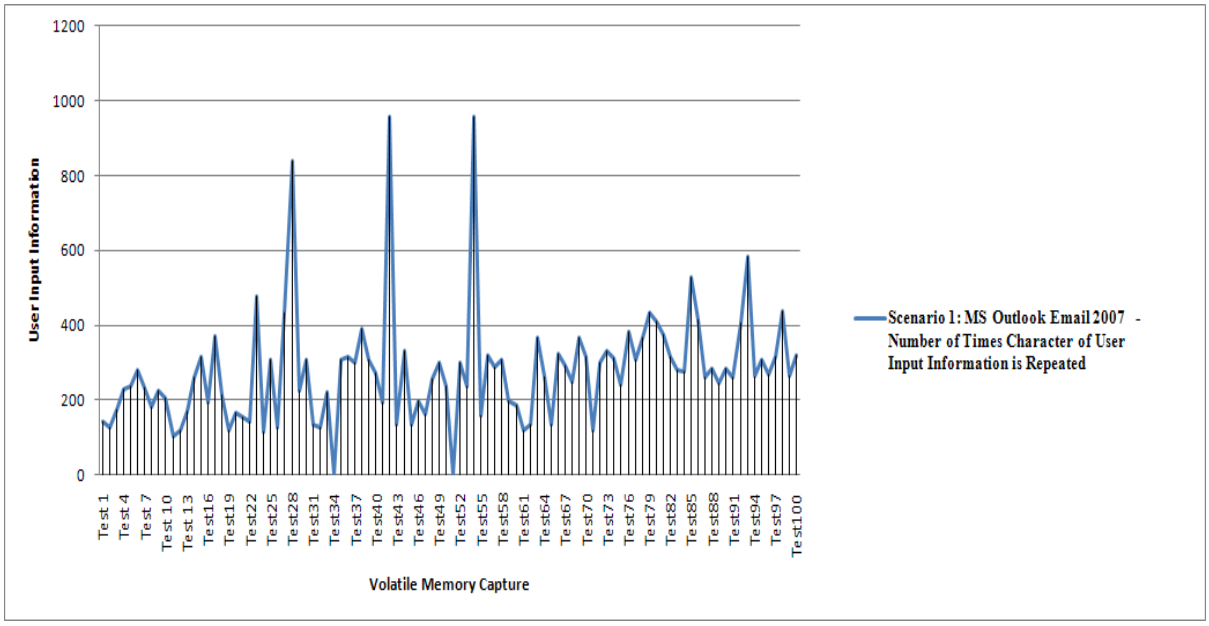


Figure 4.10 Number of times a character of user input is repeated

In Figure 4.11, the percentage of user input recovered in all tests is between 90% and 100%. The amount of data recovered exhibits a weak positive correlation with the length of user input.

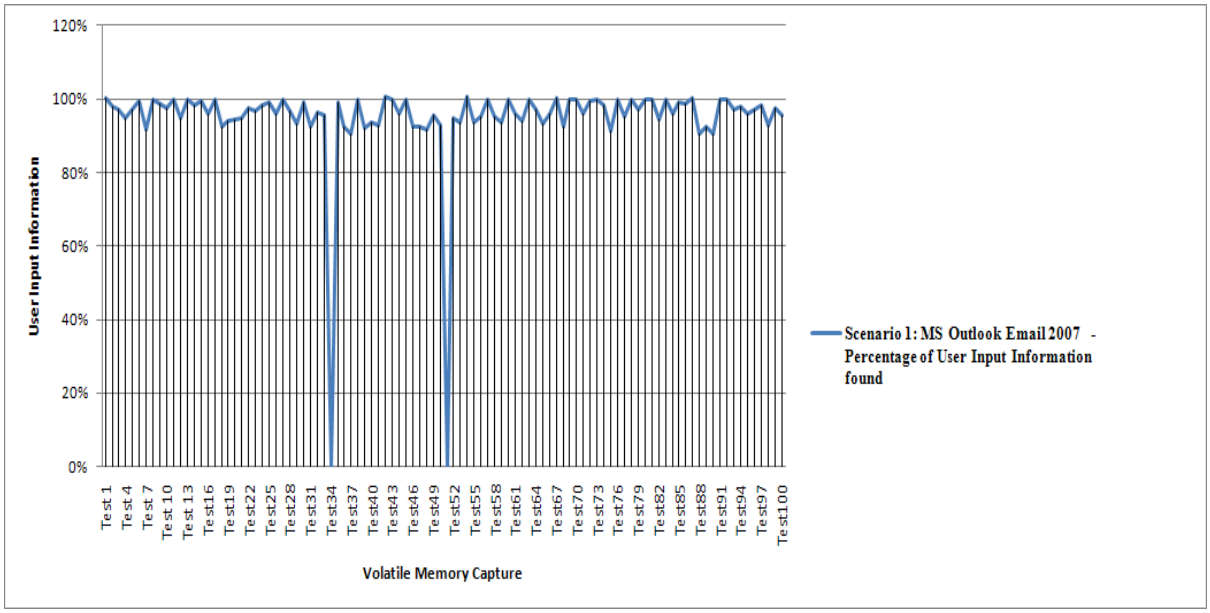


Figure 4.11 Percentage of user input found

Figure 4.12 illustrates the user input found in continuous block of the application memory. The recovery of user input is easy because the application uses UTF-16 format for their internal data representation. For example, test22, test27, test35 and test68 resulted in the highest mean lengths of continuous blocks of information found. It can be said that there are more of the initial user input made on these experiments.

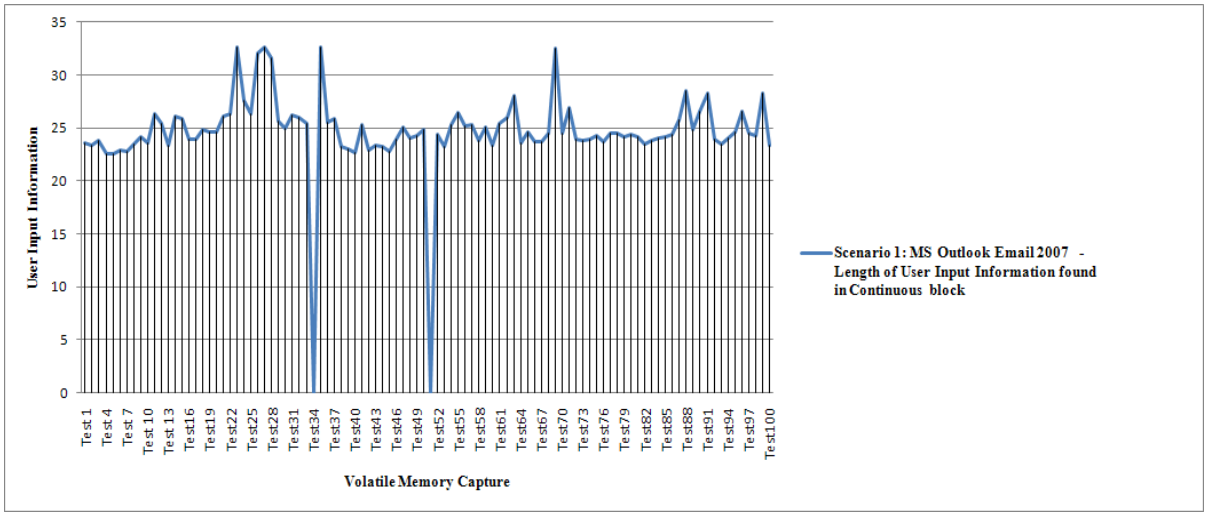


Figure 4.12 Length of user input found in continuous block

When investigating the amount of user input found using the technique of commonly used English words, it was discovered that 88% of user input appears in whole fragment and are stored in continuous blocks of the application memory. The search pattern of when user input is known resulted in whole and partial fragment of user input with 95% of related data recovered from the memory. Figure 4.12 illustrates the user input found in continuous block of the application memory. Test35 and test68 resulted in the highest mean lengths of continuous blocks of information found.

4.2.6. MS Access

As required in Scenario 1, the quantity of user input recovered from the physical memory is presented in graphs. Recovering data is very difficult on this application. In this experiment, it is difficult to identify the user input against the in-built system defined data. This is because there is a large amount of textual in-built system-defined data that resides in the application memory (See Appendix C). Three metrics are presented.

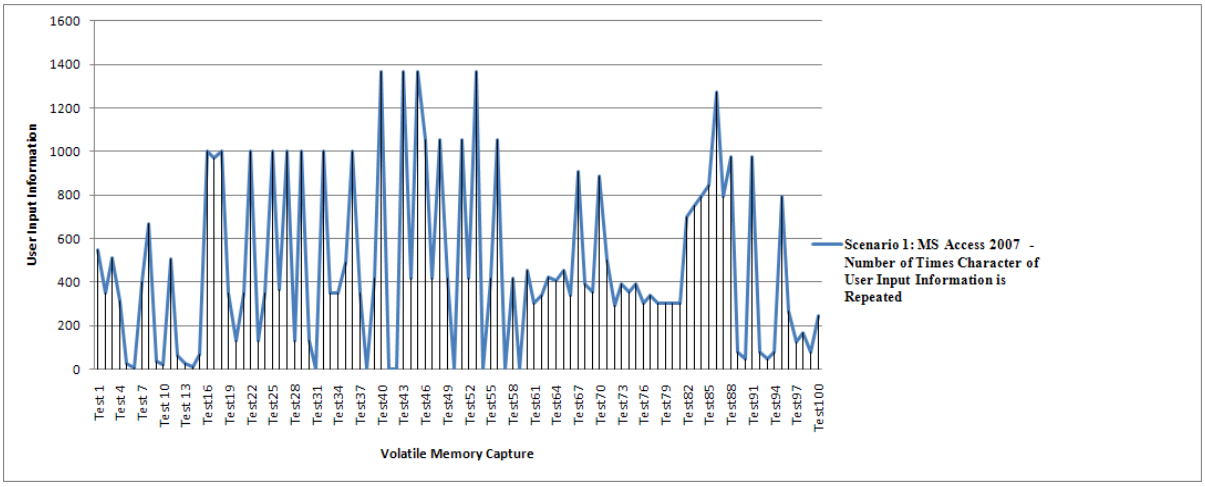


Figure 4.13 Number of times a character of user input is repeated

In Figure 4.13, test40, test43, test45 and test53 reported the highest number of repeated characters of user input found. Other recorded the lowest number of repeated character because the user input recovered revealed that there are more of contextual system in-built information than the original user input found.

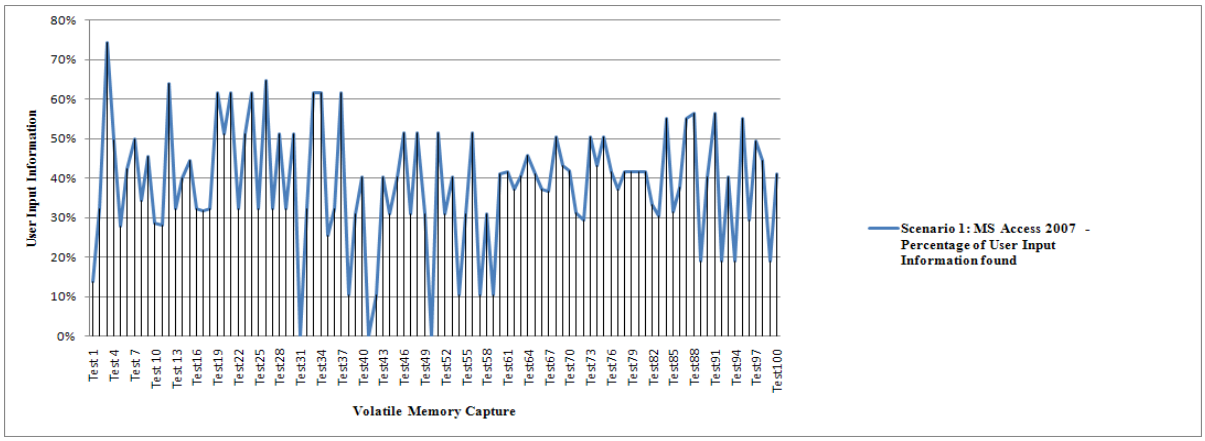


Figure 4.14 Percentage of user input found

In Figure 4.14, the highest percentage of user input found was recorded for test3, which is above 70%. In Figure 4.15, test48 and test53 proved to be at the highest for information found in continuous block. Again, there was no correlation found between the overall length of user input and any of the three metrics presented here.

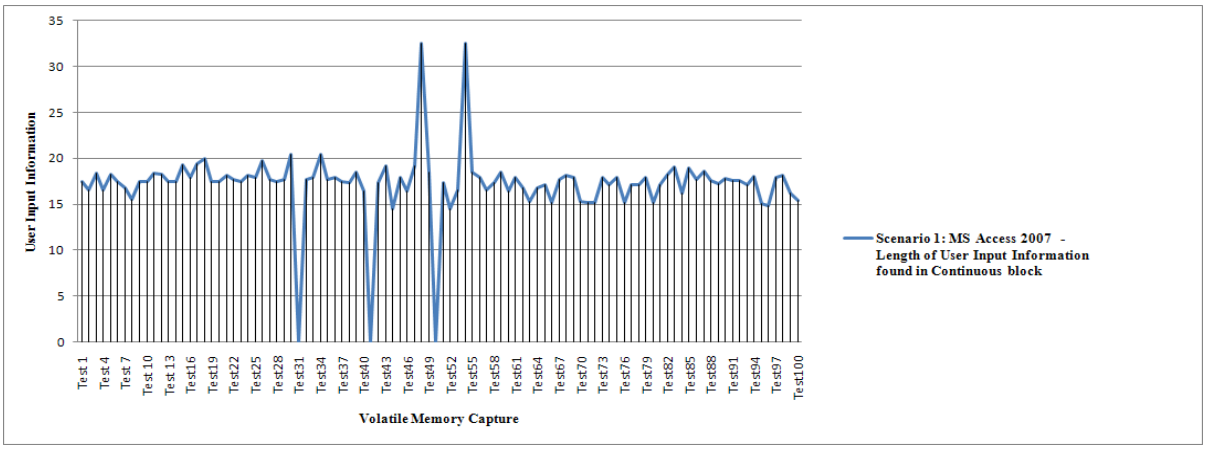


Figure 4.15 Length of user input found in continuous block

When the search pattern of when the original user input was used, 39% of related user input was found stored in the application memory whereas 30% of user input was recovered when the pattern of commonly used English words was used.

4.2.7. MS PowerPoint

In Scenario 1, PowerPoint application was investigated to identify useful information related to user input in the memory. Three plotted graphs are illustrated below. In Figure 4.16, test34 recorded the highest number of times character of user input is repeated, but the least amount was recovered in test25.

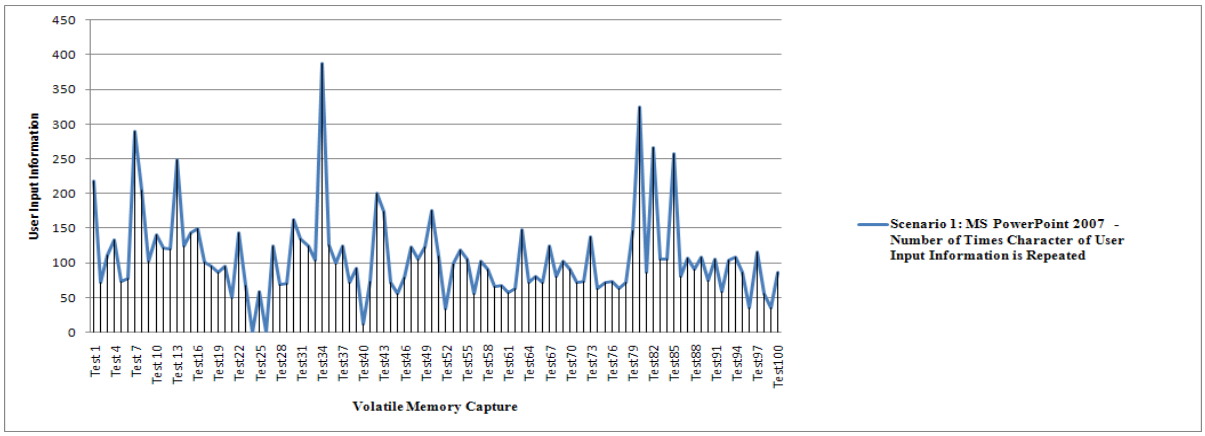


Figure 4.16 Number of times a character of user input is repeated

In Figure 4.17, the percentage amount of information found in test25 was slightly above 90%. There is a slight decrease until a sharp increase was found in test34. It was obvious that there was more user input found on this application.

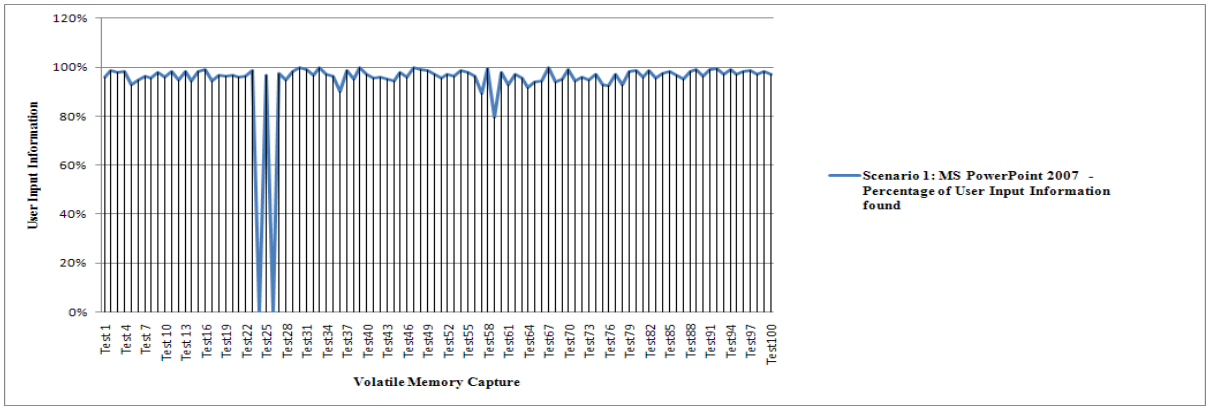


Figure 4.17 Percentage of user input found

In Figure 4.18, the result of the length of user input found in continuous block varied and test48 and test93 reported the highest peak of the information.

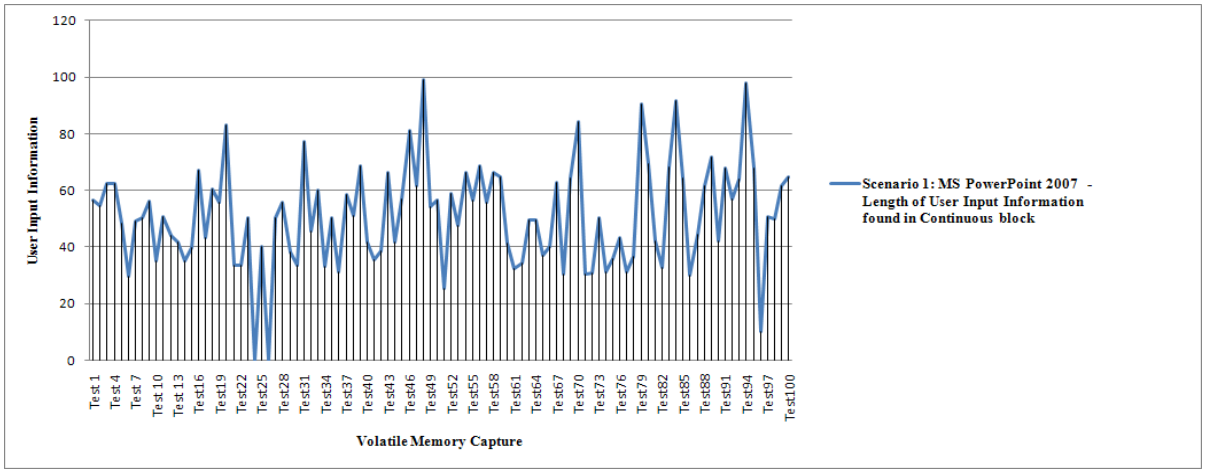


Figure 4.18 Length of user input found in continuous block

Further investigation on the search pattern techniques of known information about the original user input revealed that 95% of user input was partially stored in the allocated memory of the application. It was found that whole fragments of user input are found most and these were stored in continuous blocks of the memory whereas 87% of user input was recovered when commonly used English words was used to pattern match.

4.2.8. MS Internet Explorer 7.0

In Scenario 1, the extracted application level information of Internet Explorer was investigated and three metrics were plotted as shown below.

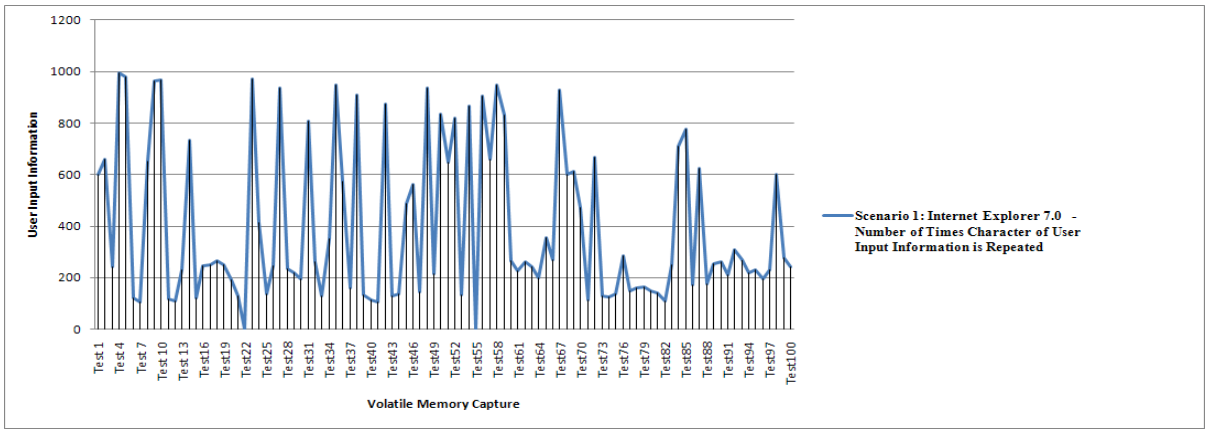


Figure 4.19 Number of times a character of user input is repeated

As shown in Figure 4.19, there are lots of examples of characters being repeated multiple times within the application memory. Figure 4.20 reported the percentage amount of information found and in all tests between 90% and 100% of user input was found in tests where user input was made.

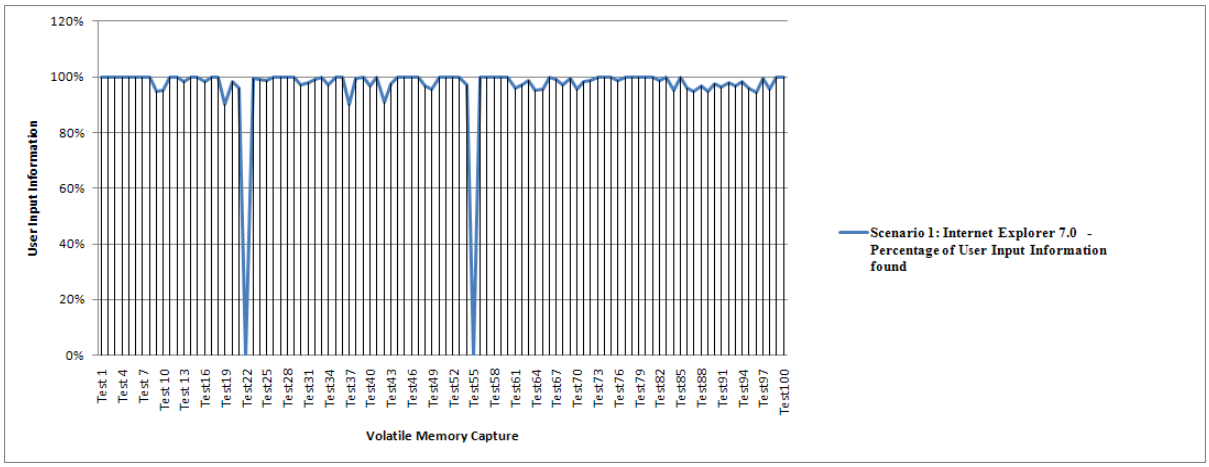


Figure 4.20 Percentage of user input found

A strong positive association between the length of user input and the percentage of user input was found. Almost all the original user input made on this application can be easily recovered from the application memory. Internet Explorer seems to allocate contiguous memory blocks to the web pages that a user browses to, making it easy to recover.

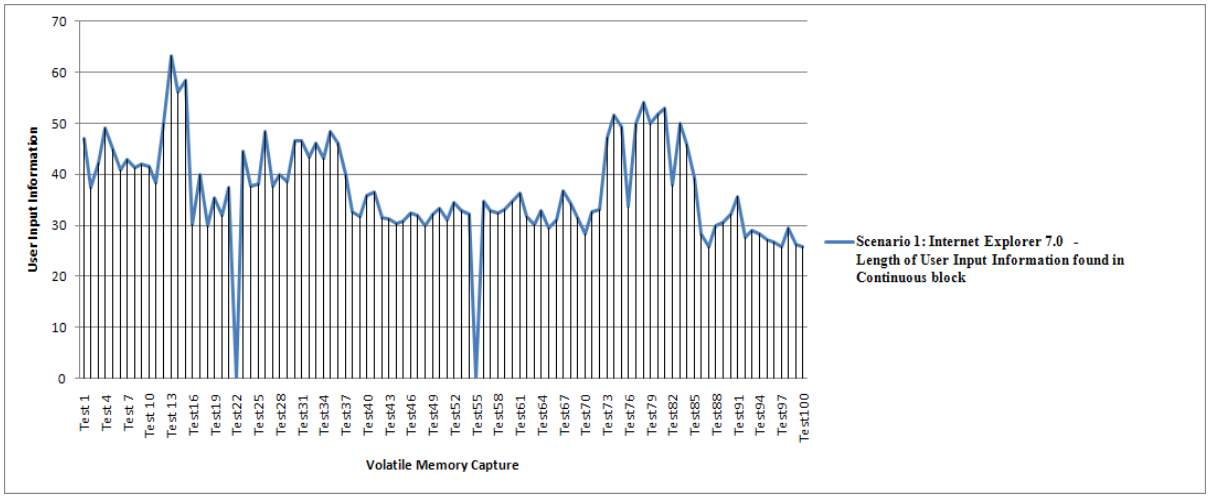


Figure 4.21 Length of user input found in continuous block

When comparing the two searching patterns, 97% of user input was recovered when known information about user input was used. As illustrated in Figure 4.21, both whole and partial fragments of user input were found stored in the memory and a large amount of user input was stored in continuous blocks of the memory and more of this information contains whole fragment of user input. When the pattern of commonly used English words was applied, 89% of user input was recovered.

4.3. Scenario 2

In this scenario, the investigation focussed on how much information is lost if the application is running but user is not interacting with the application. 100 measurements of volatile memory were captured for six days, at the beginning of each day, user input was made once on applications and no other inputs were made but images were captured at interval of 30 minutes.

4.3.1. Length of user input

In this Scenario 2, Table 4.2 presents the length of user input on the seven most commonly used applications.

Table 4.2 Length of user input

Type of Application	Day1	Day2	Day3	Day4	Day5	Day6
Adobe Reader 8.0	458	468	478	460	469	470
MS Word 2007	890	895	435	568	328	373
MS Excel 2007	253	369	281	549	287	139
MS Outlook Email 2007	641	803	951	369	533	550
MS Access 2007	107	303	287	586	242	469
MS PowerPoint 2007	639	427	437	371	401	309
MS Internet Explorer 7.0	947	876	637	417	530	359

4.3.2. Adobe Reader 8.0

In this scenario, the recovery of user input on Adobe Reader 8.0 was investigated to identify the amount of data stored in the memory as shown in Figure 4.22.

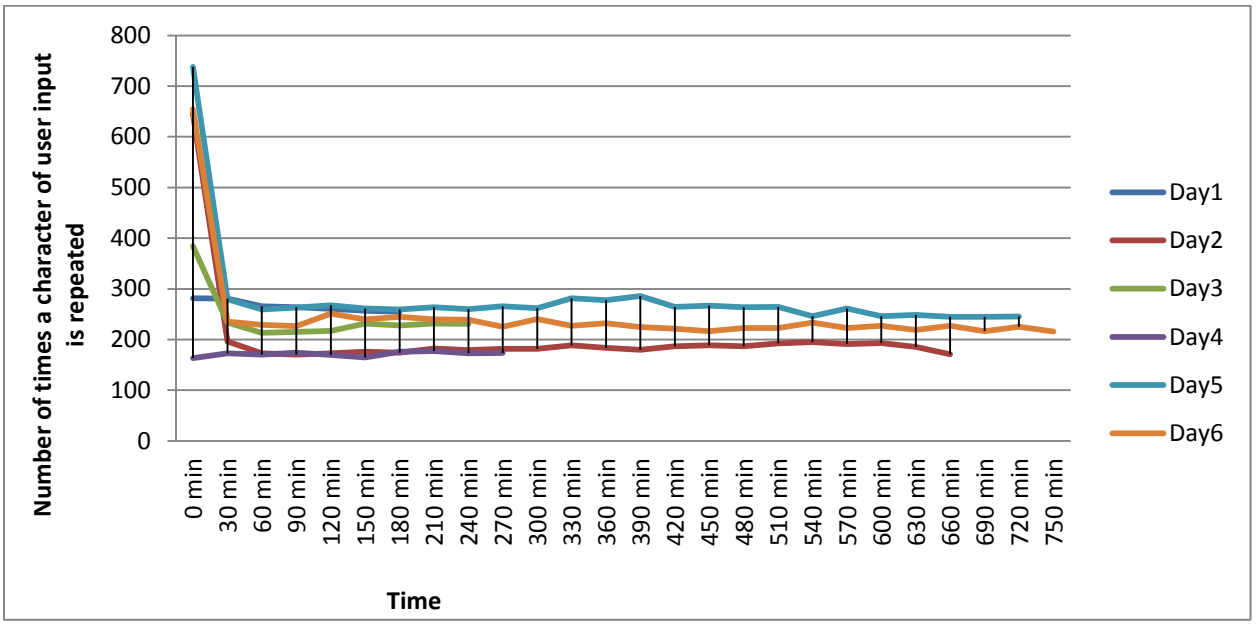


Figure 4.22 Number of times a character of user input is repeated

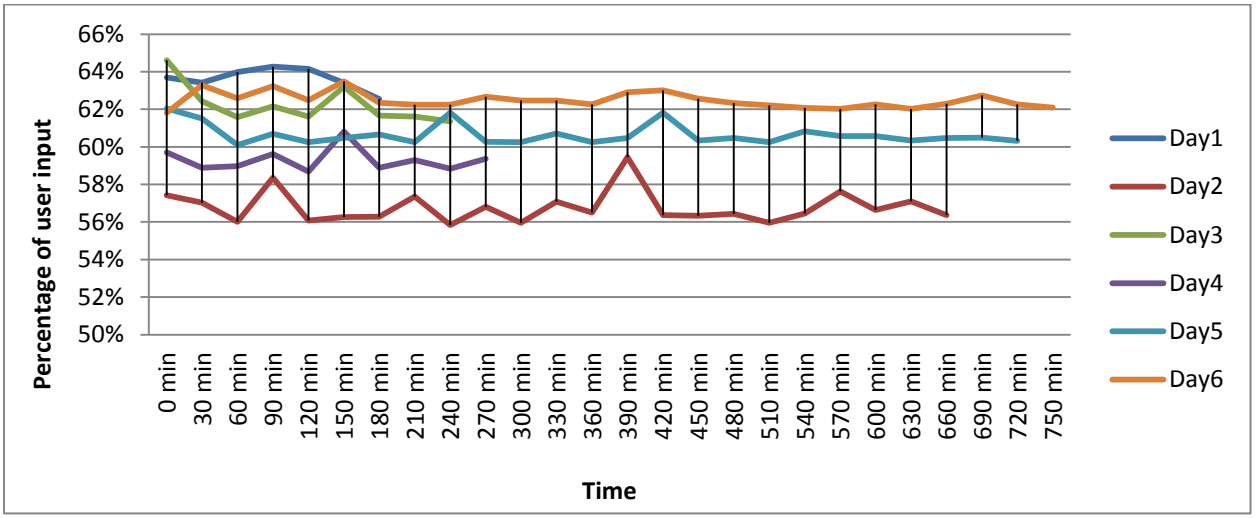


Figure 4.23 Percentage of user input found

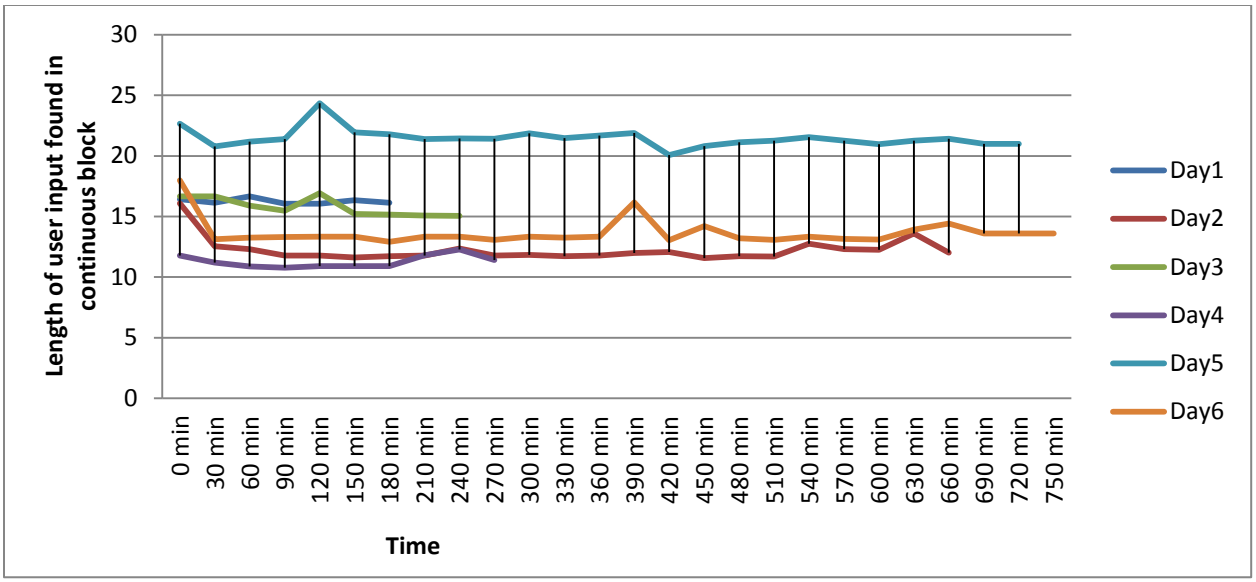


Figure 4.24 Length of user input found in continuous block

Figure 4.22 illustrates the percentage of user input that is retained in the memory. It proved to be very easy to identify the highlights and searches made on this application because of the existence of related user input in the memory. Figure 4.23 shows that user input is stored in the allocated memory and it can be retained in the memory for a long period of time. In Figure 4.24, it was found that there is strong positive correlation coefficient between the length of user input with the number of times a character of user input is repeated while weak positive correlation coefficient was found between the length of user input and the information found in continuous blocks. Further investigation into the search techniques revealed some vital information when comparing the previous result found in scenario 1. When using commonly used English words, it resulted that 48% of user input was found in the memory allocated to this application whereas, when known information about user input is used, 61% of user input was found.

4.3.3. MS Word

In this scenario, the recovery of user input on Word was investigated to identify the amount of data stored in the memory. One metric is shown below. There are small variations on the amount of relevant user input recovered in the allocated memory between each day.

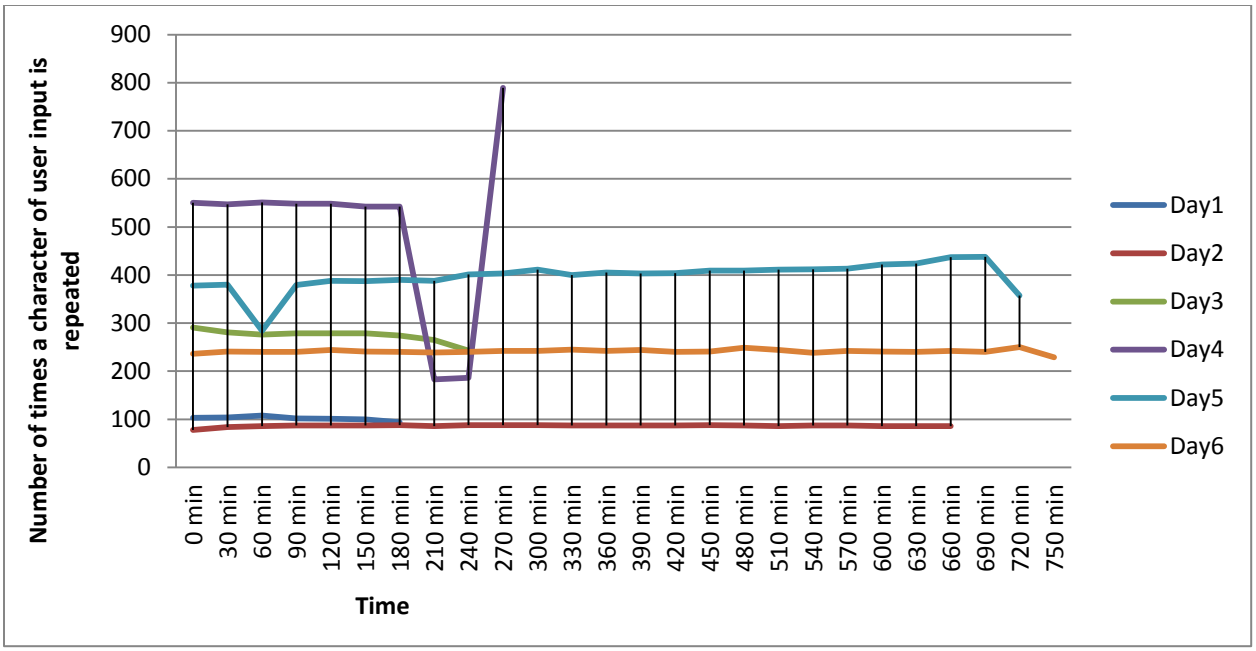


Figure 4.25 Number of times a character of user input is repeated

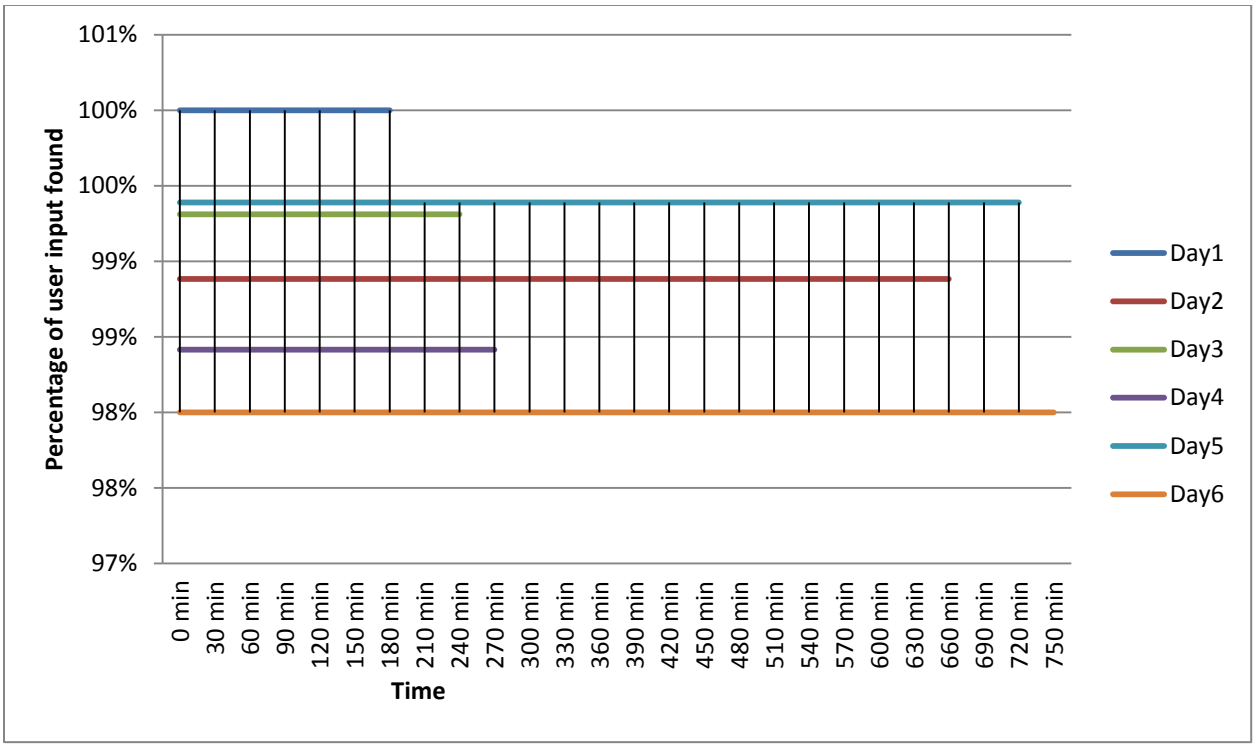


Figure 4.26 Percentage of user input found

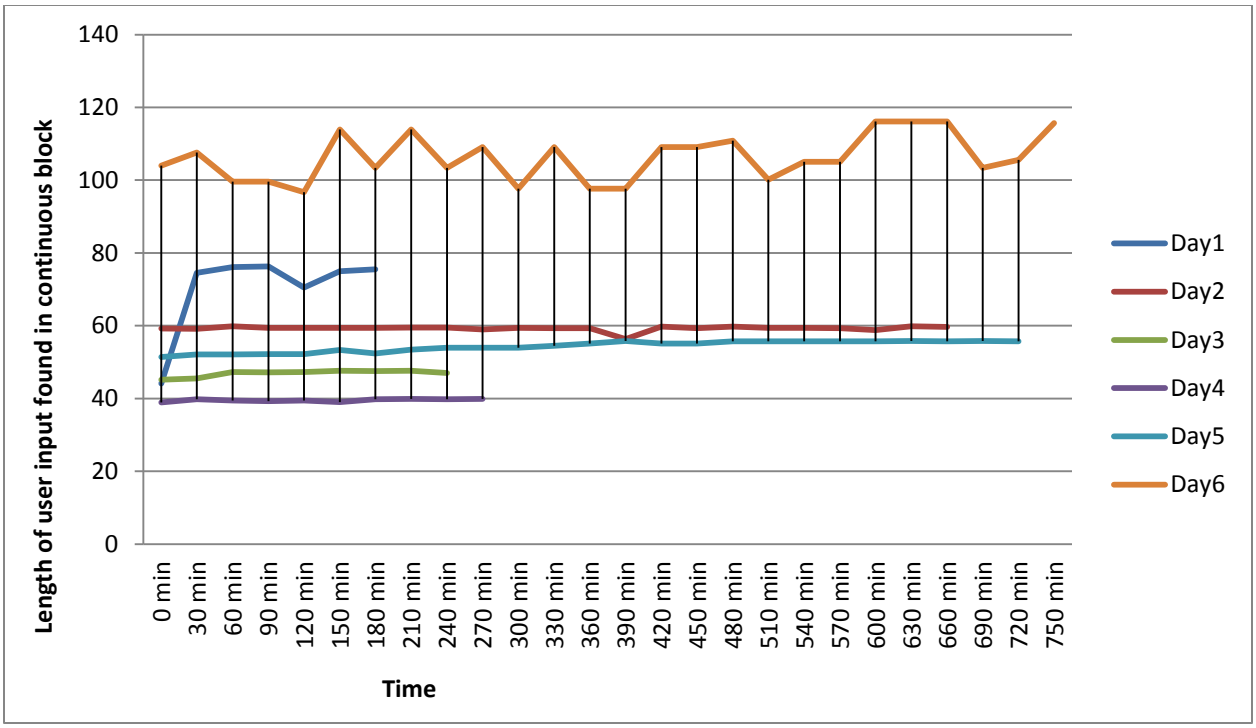


Figure 4.27 Length of user input found in continuous block

As shown in Figure 4.25, in day4, there is sharp increase in the number of times a character of user input is repeated. This could be the result of some system in-built information that was recovered as stored in the memory. What is also noticeable in each day of the experiments is that there is a constant amount of data recovered. The result of this experiment is contrary to the research work of (Jason, Ewa, Derek, & Magdalena, 2007) where it was stated that the majority of data in the memory are persistent for less than 5 minutes. It can be said that the data persistency in the memory allocated to MS Word application is much greater than this. Overall, the percentage of user input found in Figure 4.26 was between 98% and 100%. In Figure 4.27, the length of user input found in continuous block was unstable in day6. This could be the result of some in-built system information that was stored on the application memory. In this scenario, investigation into the techniques used indicated that a large amount of user input is stored in the memory and this information was retained for a long period of time in the memory. When comparing the two search patterns of user input activities on application memory, it was discovered that 99% of user input was found when known information about user input was applied whereas, 87% of user input was found when using pattern of some commonly used English words.

4.3.4. MS Excel

In this scenario, the memory allocated to Excel was investigated to identify the amount of user input that can be recovered from the memory. The two search patterns were investigated and it was discovered that when known information about user input was applied, 51% of related user input was found in the memory whereas, 41% of user input was found when some commonly used English words was used.

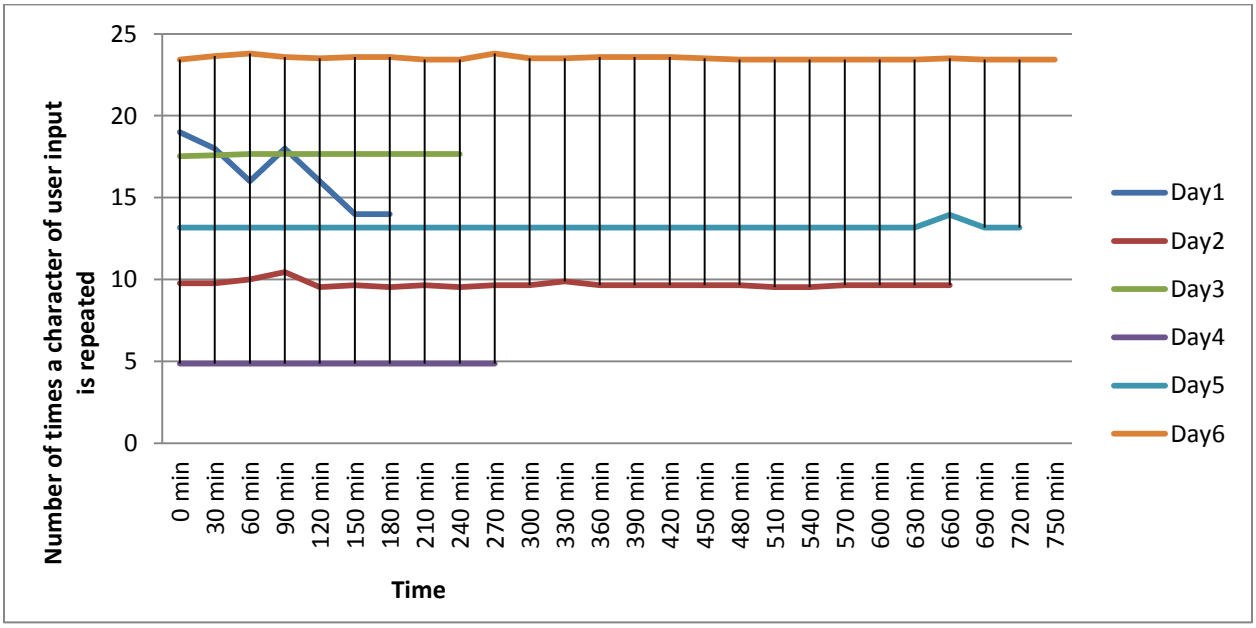


Figure 4.28 Number of times a character of user input is repeated

The amount of user input recovered was small on this application. In Figure 4.28, the number of time a character of user input is repeated dropped sharply at 60minutes in day1 and rise up again until it was found at its lowest in 180minutes. It was found that the system in-built numeric character was found more than the user input. In Figure 4.29 below shows that the recovery of data in the memory is found to be difficult in some cases. The percentage amount of user input found is different at each day of the captures. For example, in day4, the amount of user input found was the lowest. It was found that little amount of text and numeric character of user input was made on this application. But the result changed in Day6. For example in Day6 there is a large amount of data found. This is because the characters of user input consist of more text than numeric characters and where more of numeric characters were made than the text character, the

recovery of data was more difficult because of the existence of textual in-built system data stored in the application memory (See Appendix B).

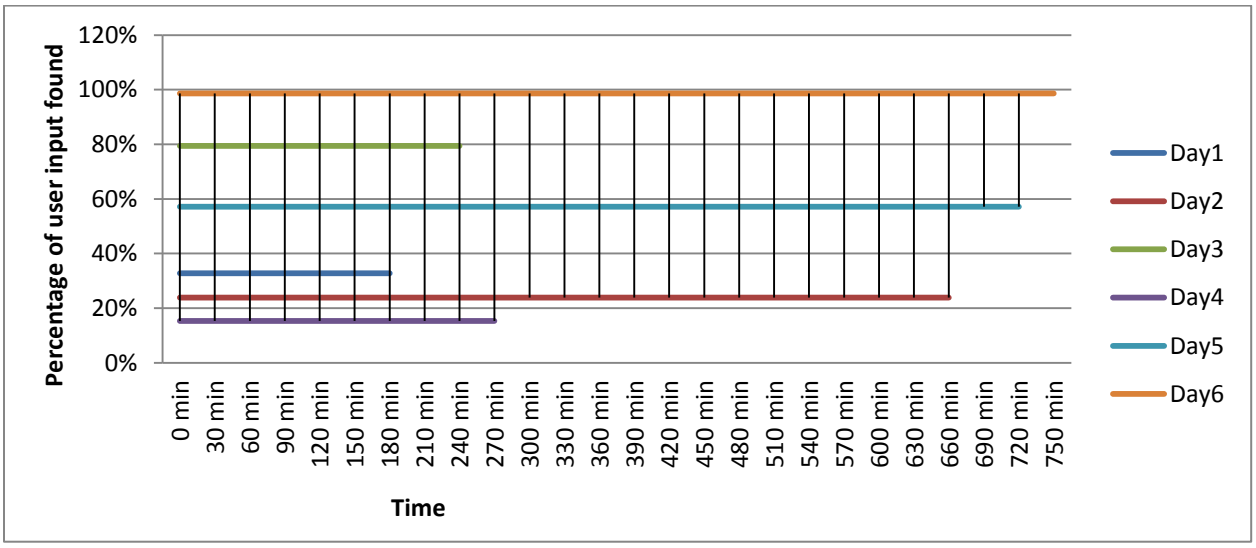


Figure 4.29 Percentage of user input found

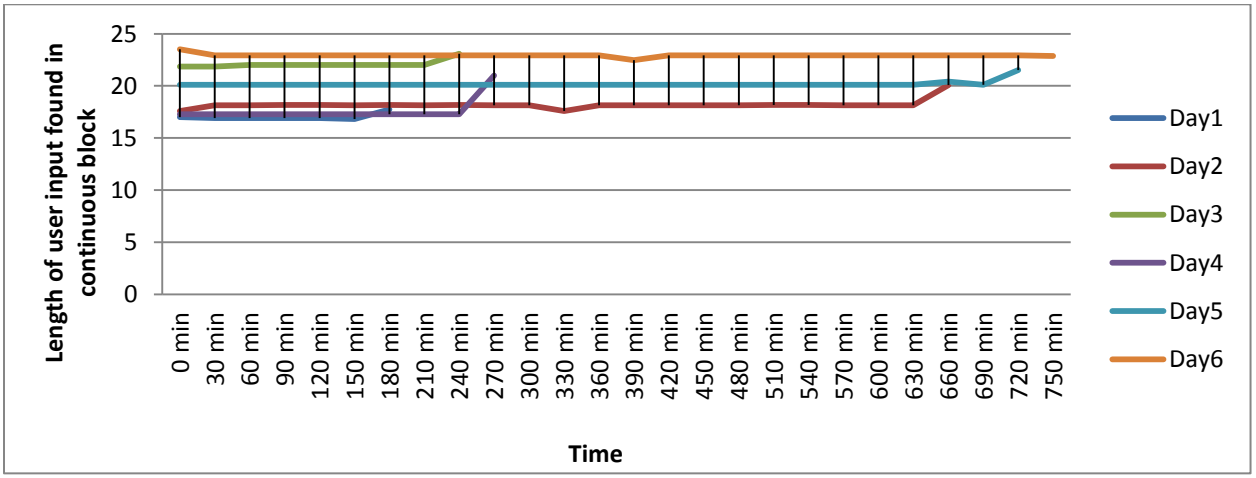


Figure 4.30 Length of user input found in continuous block

In Figure 4.30, the length of user input found in continuous block was at the lowest in Day1 while the highest amount of user input found in continuous block was found in Day6. When comparing the amount of user input found with the previous scenario 1 there is a slight increase in the percentage of user input found.

4.3.5. MS Outlook Email

In this scenario, the recovery of user input on Outlook was investigated to identify the amount of information stored in the memory. Recovery of data is easy on this application because of the large amount of user input that is stored in the allocated memory.

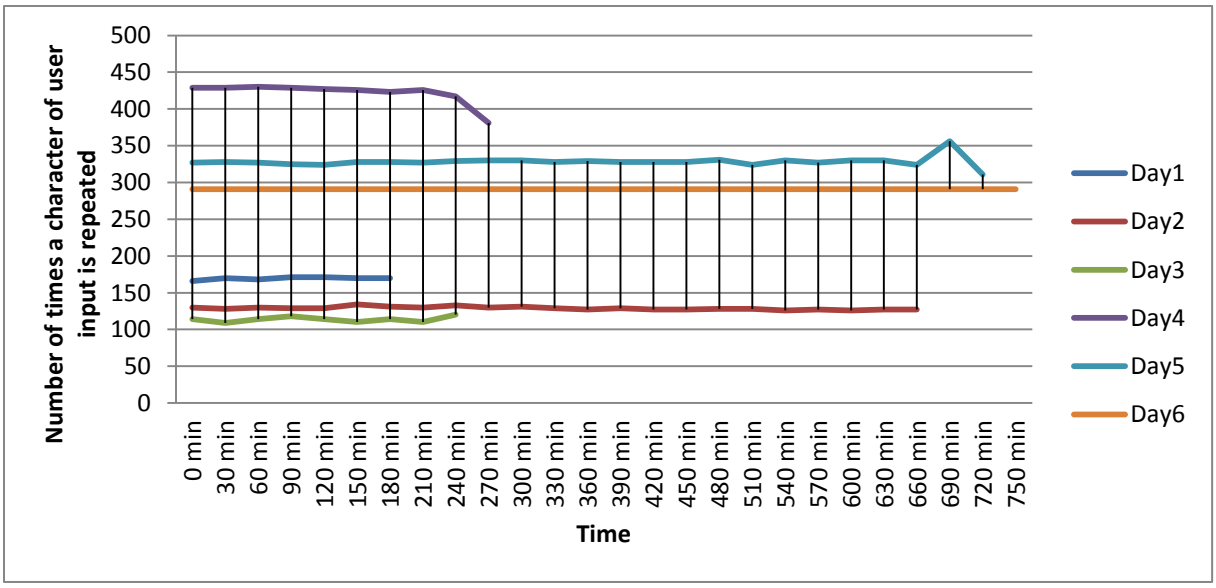


Figure 4.31 Number of times a character of user input is repeated

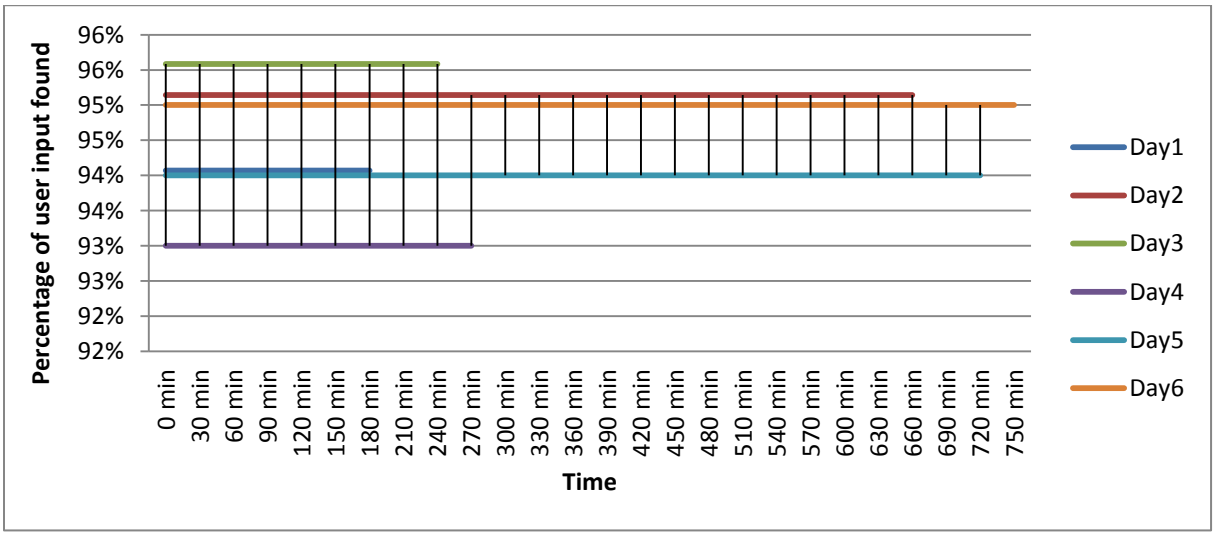


Figure 4.32 Percentage of user input found

When the two search patterns were compared, it was discovered that 97% of user input was stored in the memory when known information of the original user input was applied, whereas 75% was stored in the memory when the pattern of commonly used English words was used. It can be said that user input can be stored for a long period of time in memory.

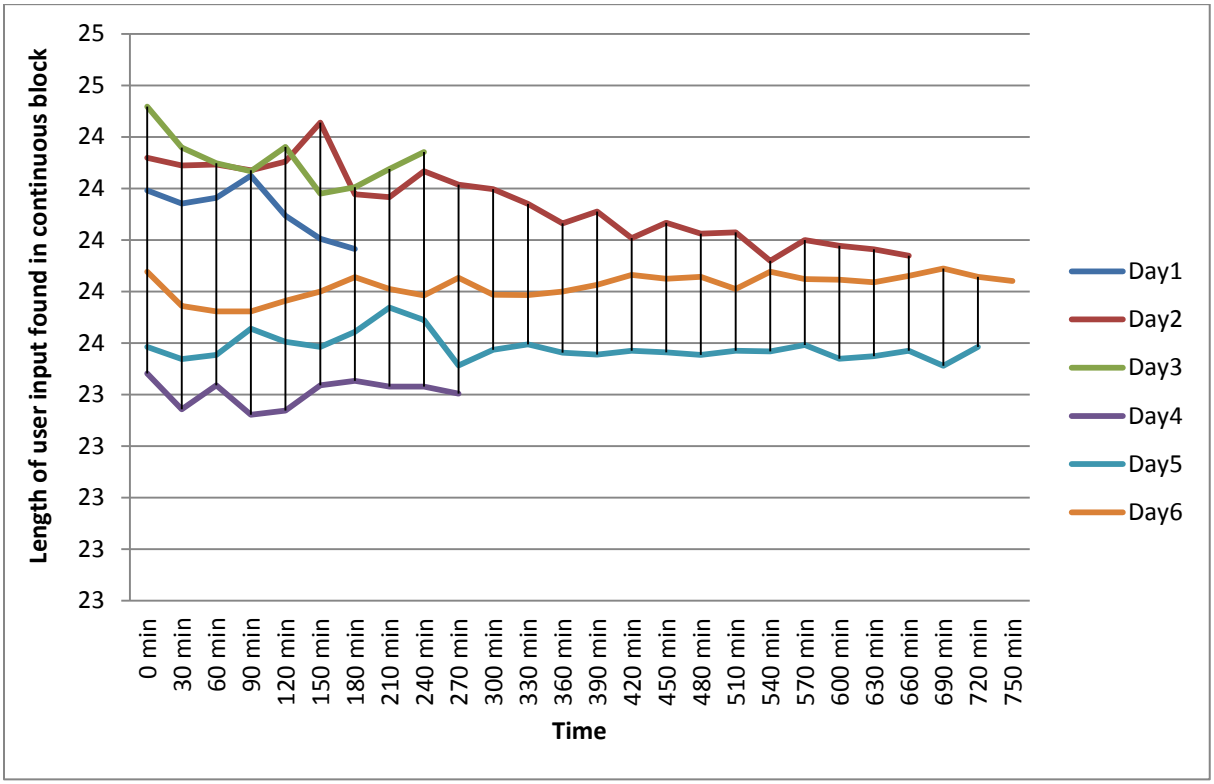


Figure 4.33 Length of user input found in continuous block

The graph shown in Figure 4.31 illustrates the number of times a character of user input is repeated and as shown, in Day4, a large amount of data was repeated in the memory. Figure 4.32 illustrates the percentage of user input and the amount of user input found was between 93% and 96%. Figure 4.33 indicates that the user input may be found in contiguous blocks of memory, making it easier to recover user input. User input sent and received are retained for a longer period of time in memory which makes finding the data easier.

4.3.6. MS Access

Following the research requirement in Scenario 2, the application level information was recovered from MS Access application. Figure 4.34 illustrates the number of times a character of user input is repeated, in Day 1, the amount of user input rise up to the highest. It was found that there is little user input and more of the system in-built information was recovered as stored in the application memory.

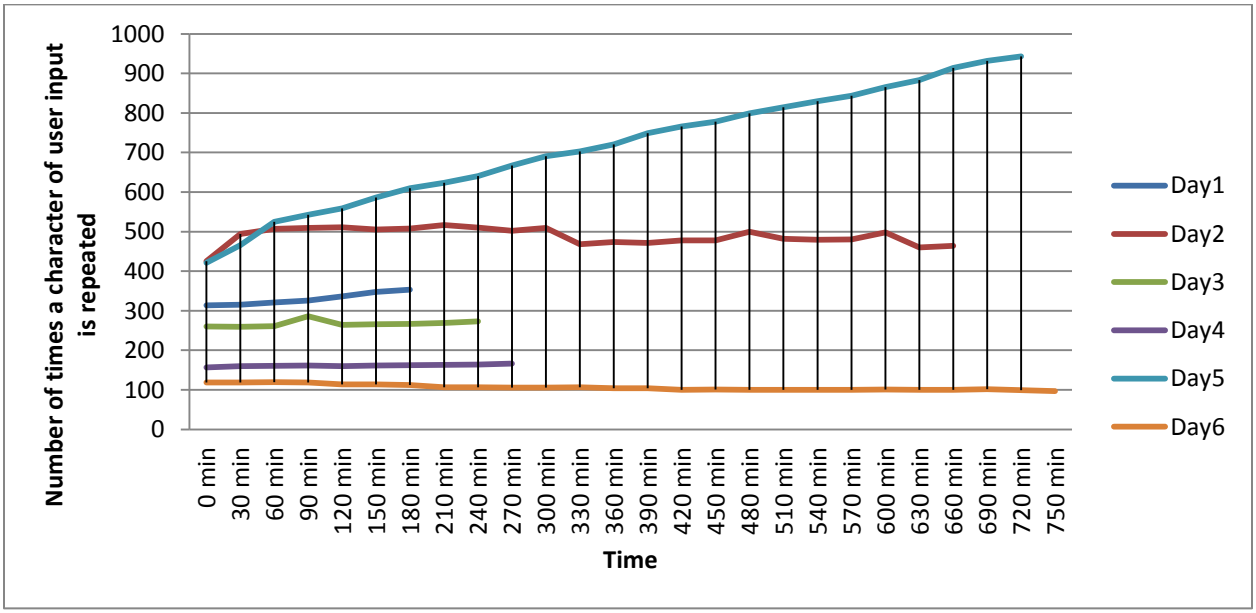


Figure 4.34 Number of times a character of user input is repeated

Figure 4.35 illustrates the percentage of data stored over time. The percentage of user input found was between 45% and 65%. This shows the retention of user input found on the application memory. For example, in Day5 user input is reduced. Although, the in-built system defined data of the application appears more often in the memory than the user defined input, there is a slight change in the result found in the previous scenario. It can be said that user input is partially stored and dispersed in the memory allocated to this application.

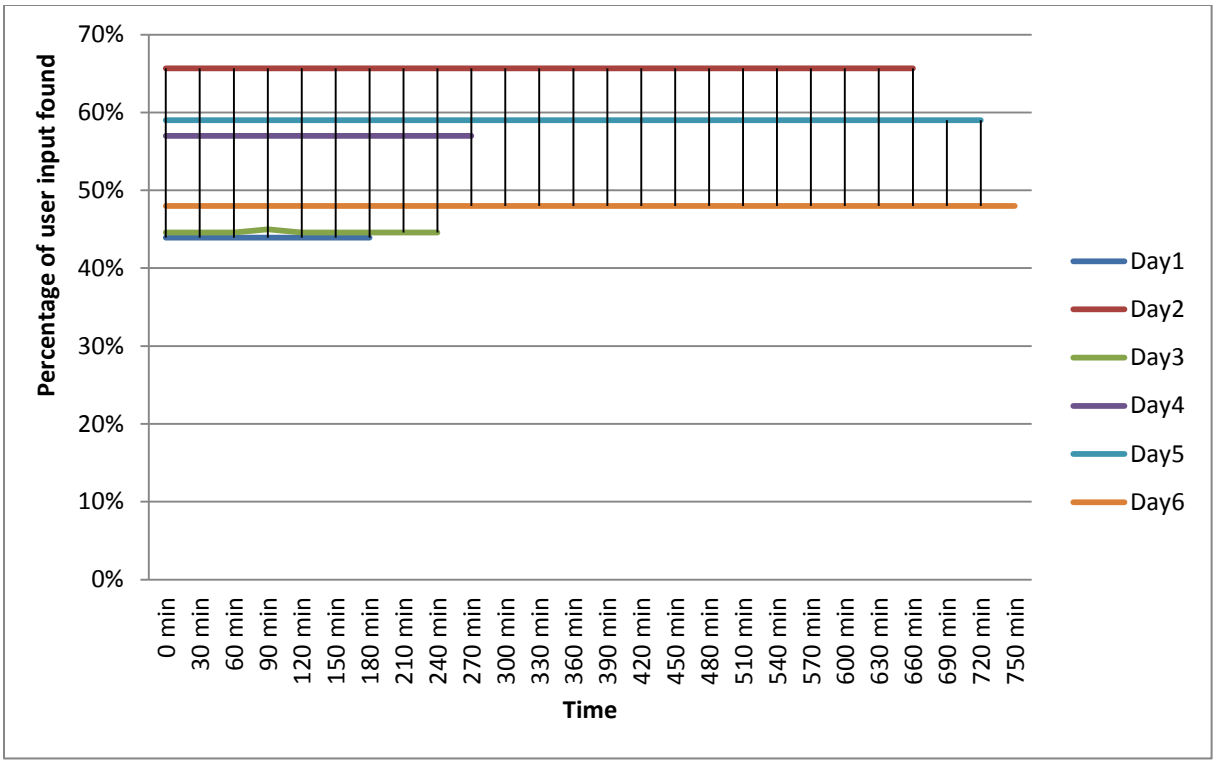


Figure 4.35 Percentage of user input found

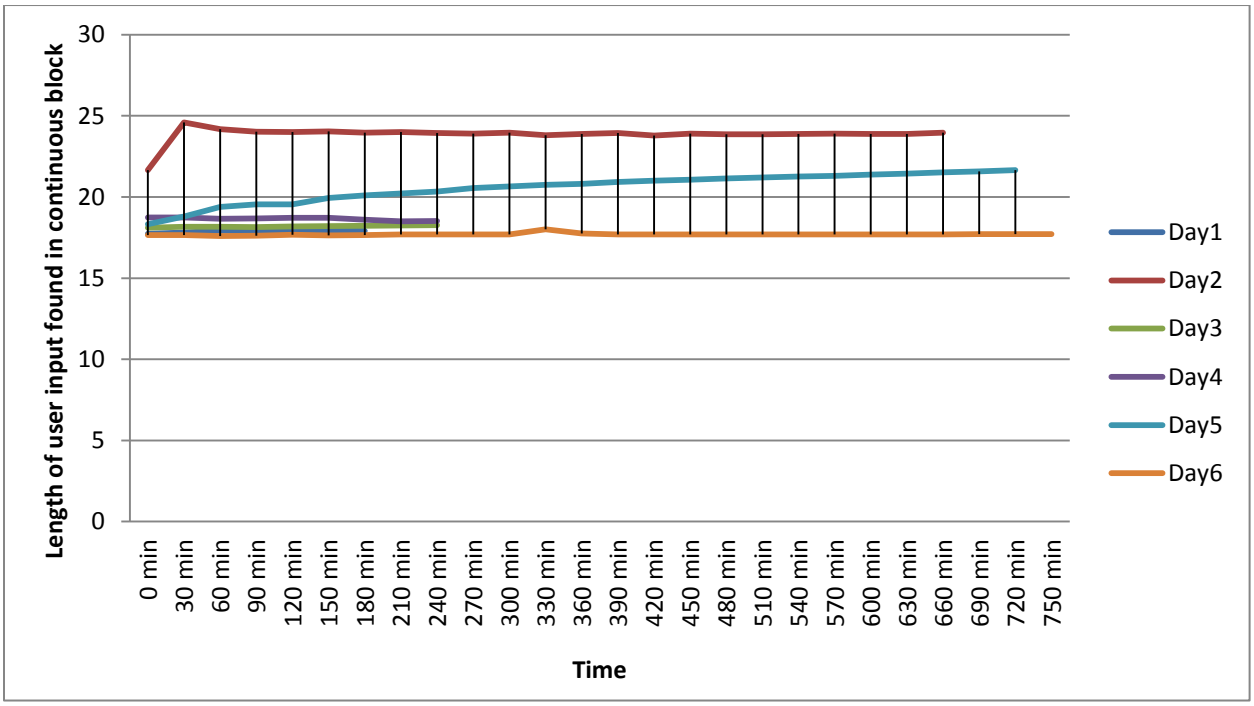


Figure 4.36 Length of user input found in continuous block

The recovery is difficult because of the system-defined in-built data in the memory (See Appendix C). Obviously, Day1 of Figure 4.36 shows the lowest amount of the length of user input found in continuous block. When pattern searching of the original user input was used, 48% of related data were found stored in the memory whereas, only 10% of user input was found when pattern of some commonly used English words was applied.

4.3.7. MS PowerPoint

In Scenario 2, PowerPoint application is investigated to identify user input stored in the memory allocated to this application. The technique used to search for user input was investigated. Figure 4.37 illustrates the number of times a character of user input is repeated in the memory. In day4, at 30minutes, the number of repeated characters is the highest number of times a character of user input is repeated in the memory.

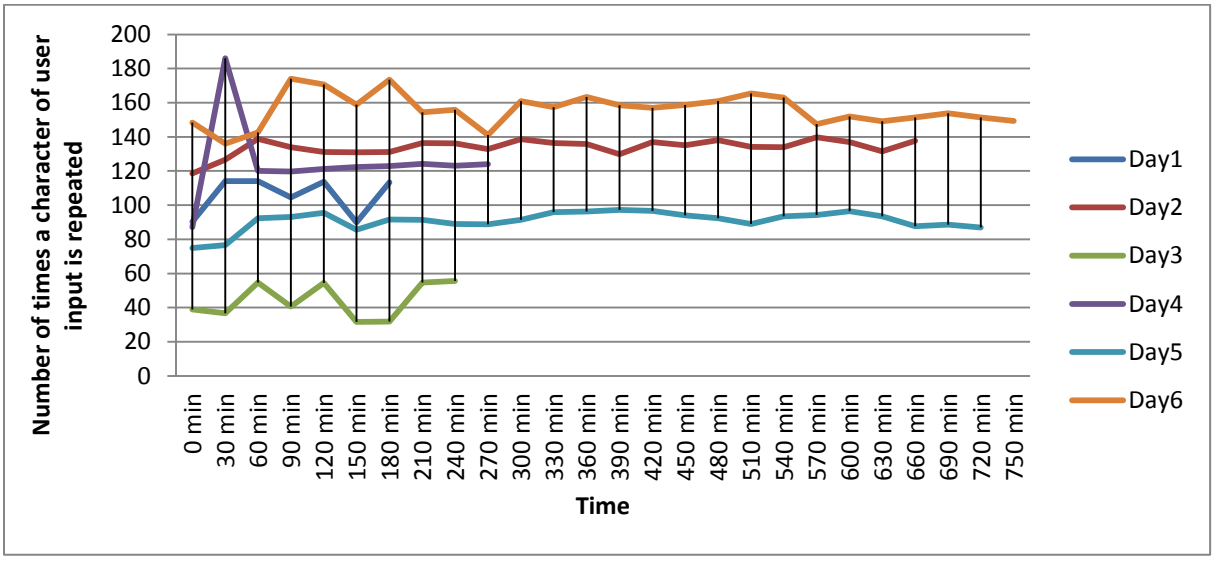


Figure 4.37 Numbers of times a character of user input is repeated

The length of user input are strongly correlated with the number of times character of user input is repeated. It can be said that user input found contains more Latin characters that was originally entered on the application.

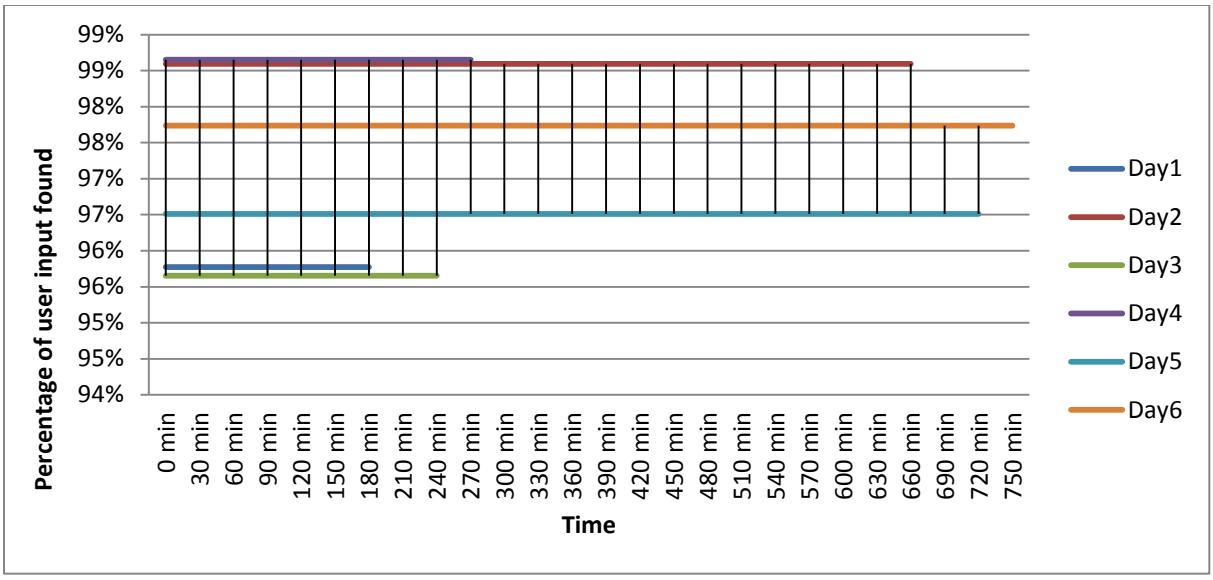


Figure 4.38 Percentage of user input found

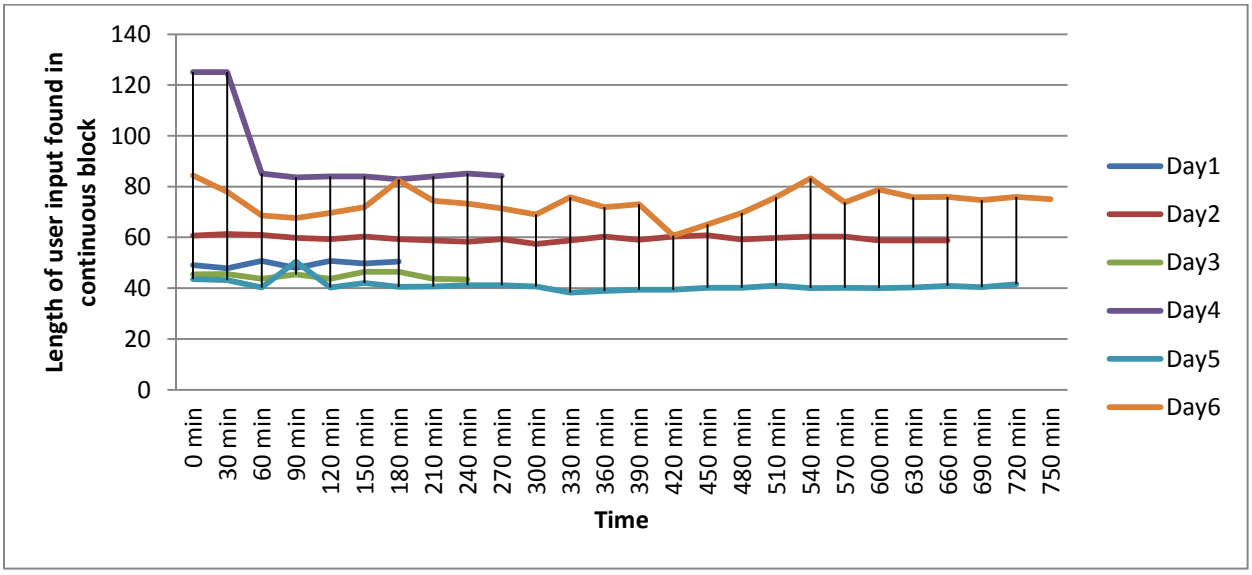


Figure 4.39 Length of user input found in continuous block

In Figure 4.38, the percentage of user input found is between 96% and 99%. The user input recovered was stored over time and was slightly increased than the result found in the previous scenario.

As shown in Figure 4.39, the length of user input found in continuous block was recorded at the highest in Day4. When pattern searching with the original user input, it was discovered that 97% of related data was found, whereas only 50% of user input was found when the pattern of commonly used English words was applied. In this experiment, it can be said that partial and whole fragment of user input are stored more in the memory. It can also be said that user input can be retained for a long period of time in the memory. Three metrics of graphs are presented.

4.3.8. MS Internet Explorer 7.0

As required in Scenario 2, an investigation was carried out on the quantity of data recovered from the physical memory of Internet Explorer 7.0. The results are presented in a graph as plotted below. Figure 4.40 illustrates the number of times a character of user input is repeated. In Day1, the user input found as repeated was at the highest.

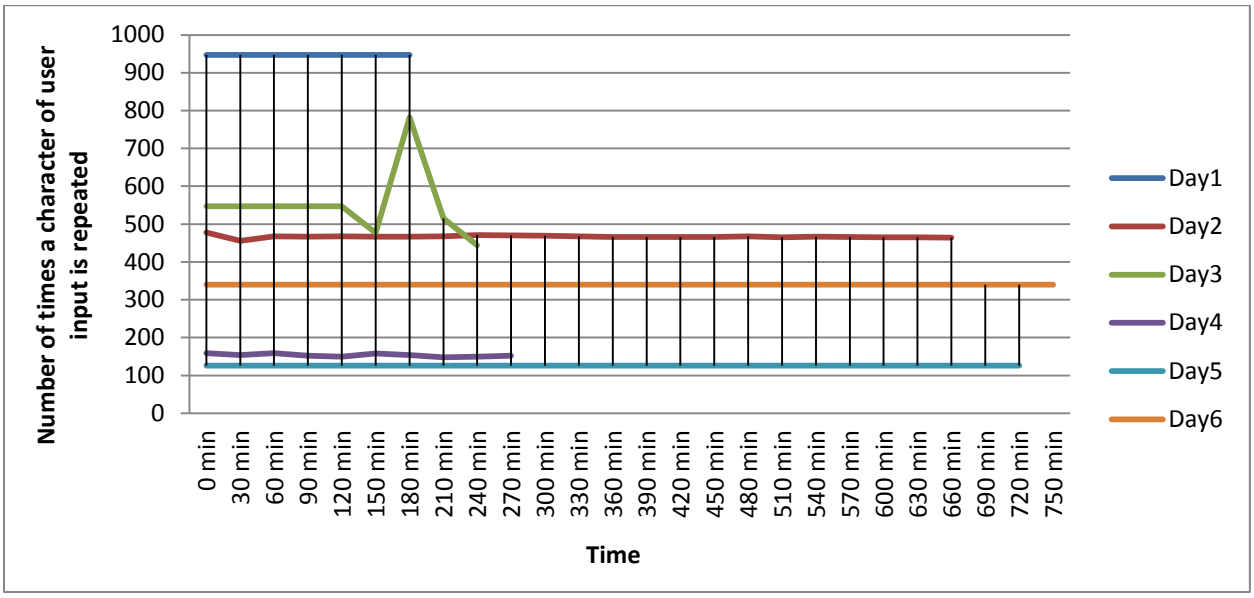


Figure 4.40 Numbers of times a character of user input is repeated

As shown in Figure 4.41, the percentage of user input found is at the lowest in Day1. However the amount of user input found is relatively large, especially when compared to other applications. It can be said that user input, such as browsing web pages, are retained for a long period of time.

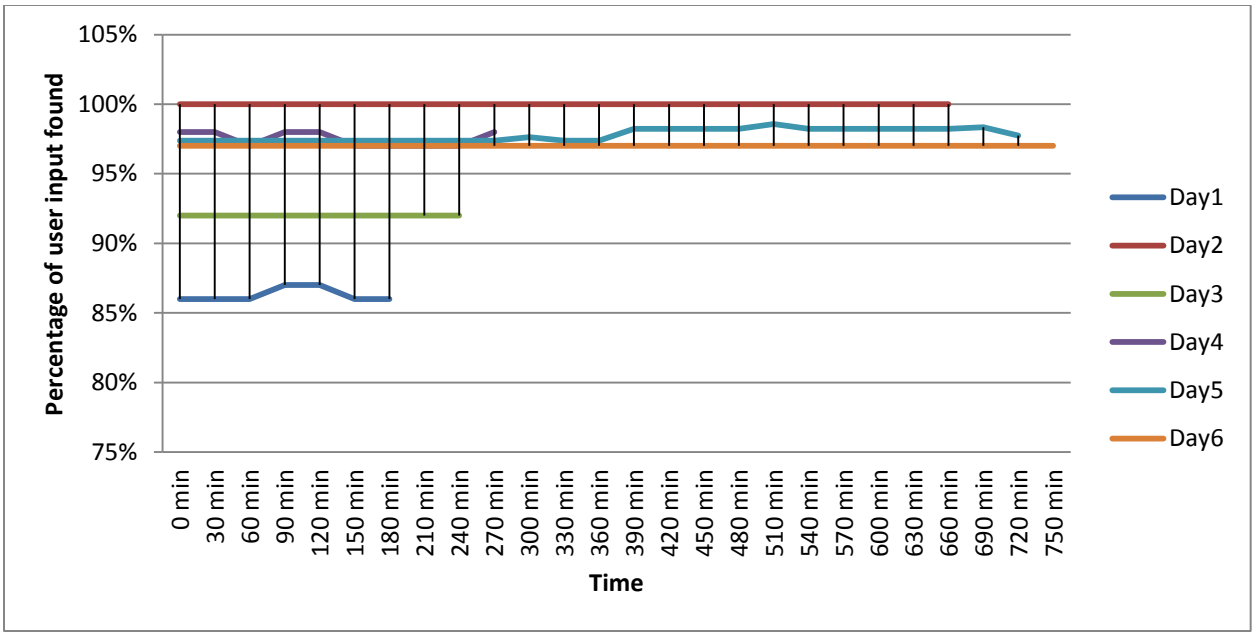


Figure 4.41 Percentage of user input found

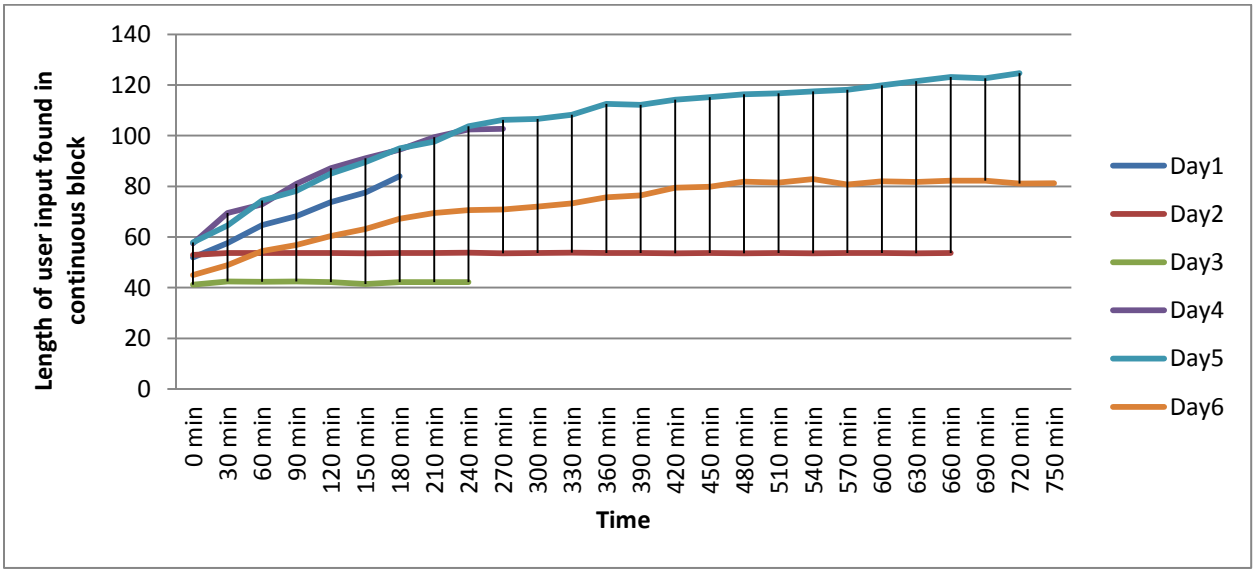


Figure 4.42 Length of user input found in continuous block

The technique used to search for user input was investigated. The length of user input found in continuous block of the application was as illustrated in Figure 4.42. For example, in Day5, the user input recovered started at the lowest and gradually moved up. When comparing the two

pattern searching techniques, it was revealed that the pattern of some commonly used English words resulted in 75% of user input found in the memory whereas, 99% of user input was found when the pattern of known information about user input was used for searching data. The user input is stored more in continuous blocks and found in whole fragments in the memory of this application.

4.4. Scenario 3

In this Scenario 3, the investigations were focused on identifying how much data is lost if the application is closed and the system is not used for any other purpose. 100 measurements of volatile memory were taken over six days. At the beginning of each day, user input was made once on the applications and the application was then closed. No other inputs were made on the computer system. The volatile memory was captured at a set interval of 30 minutes.

4.4.1. Length of user input

In this scenario 3, Table 4.3 presents the length of user input on the seven most commonly used applications. This information shows the length of user input for each day of the experiments.

Table 4.3 Length of user input

Type of Application	Day1	Day 2	Day 3	Day 4	Day 5	Day 6
Adobe Reader 8.0	468	415	853	243	249	409
MS Word	328	499	346	370	316	603
MS Excel	253	369	281	549	287	139
MS Outlook Email	281	147	187	231	307	258
MS Access	443	713	308	209	181	554
MS PowerPoint	402	257	301	476	280	307
MS Internet Explorer 7.0	990	415	768	685	717	904

4.4.2. Adobe Reader 8.0

In scenario 3, data recovery seems to be difficult with Adobe Reader 8.0, but some related user input was recovered as presented in a graph below. As shown in Figure 4.43 the lowest amount of user input was found in Day4.

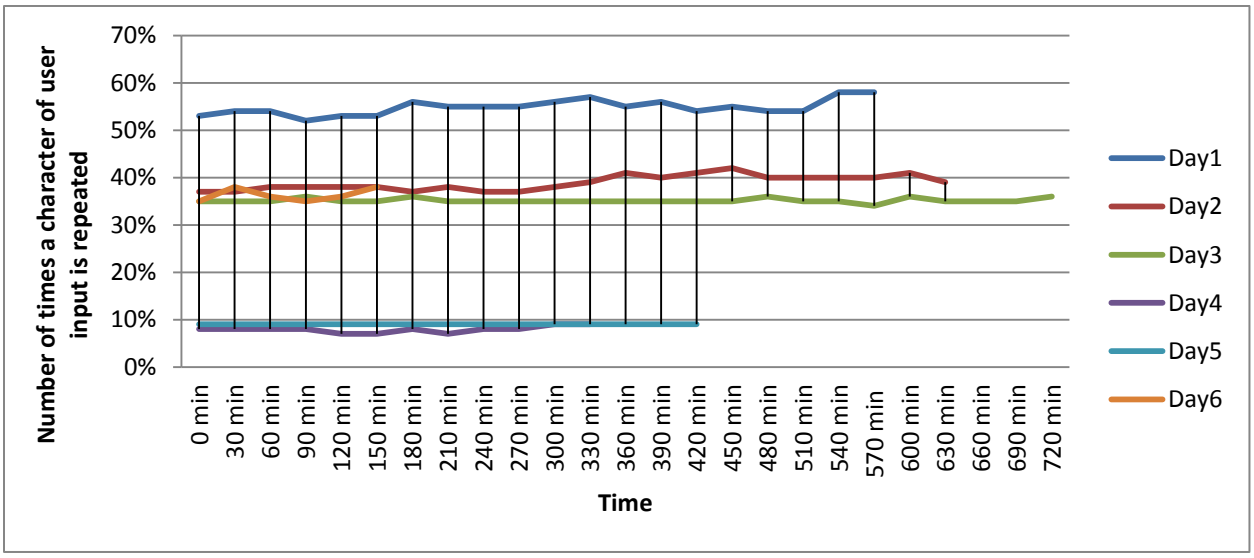


Figure 4.43 Numbers of times a character of user input is repeated

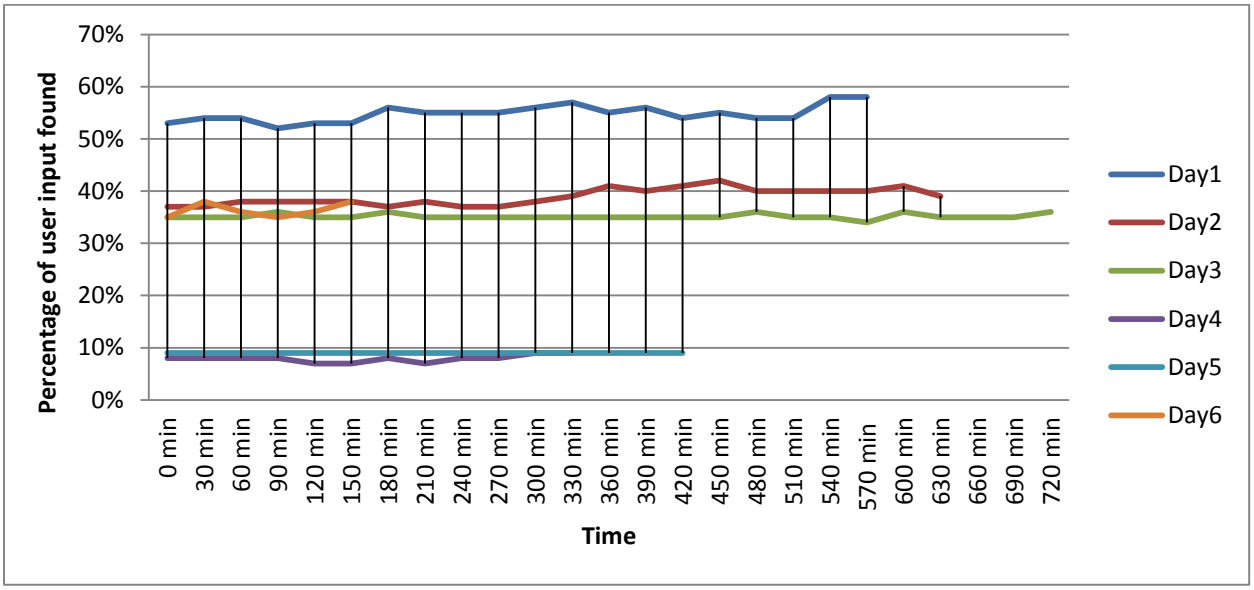


Figure 4.44 Percentage of user input found

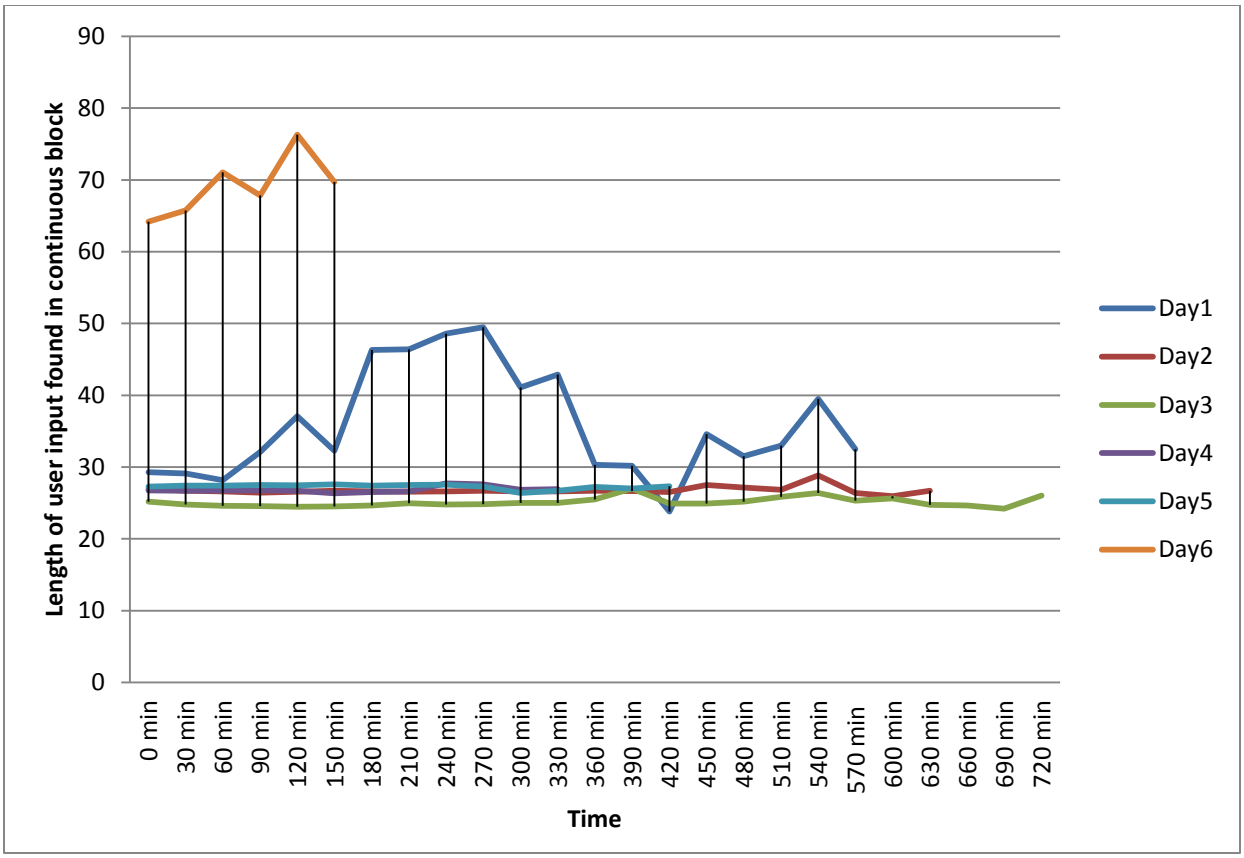


Figure 4.45 Length of user input found in continuous block

Figure 4.44 illustrates the percentage of user input that is retained in memory. It proved to be very difficult to identify the highlights and searches made on this application. Figure 4.45 shows the length of user input found in continuous block. For example in Day6, the amount of user recovered increases gradually. It was found that there is less user input on this application than the system in-built information that was found. When compared to the information recovered in Scenario 2, the percentage of user input found in Day4 of this Scenario 3 is lower than any other day of the experiments in this research project. The result of the investigation is relatively low when compared with the previous scenario. The amount of data stored over time differs when the two pattern searching techniques are investigated. The pattern of when known information about the original user input was used with 30% of related user input stored in the memory but, when pattern some commonly used English words was applied, 15% of user input is found stored in the memory. This information is partially stored and dispersed in the memory allocated to this application.

4.4.3. MS Word

As required in Scenario 3, the extracted application level information was investigated to find out the quantity of information stored over time in the volatile memory. The information recovered is presented in a graph below. Figure 4.46 illustrates the number of times a character of user input is repeated. In Day6, there was little user input found while in Day1, the user input found was large.

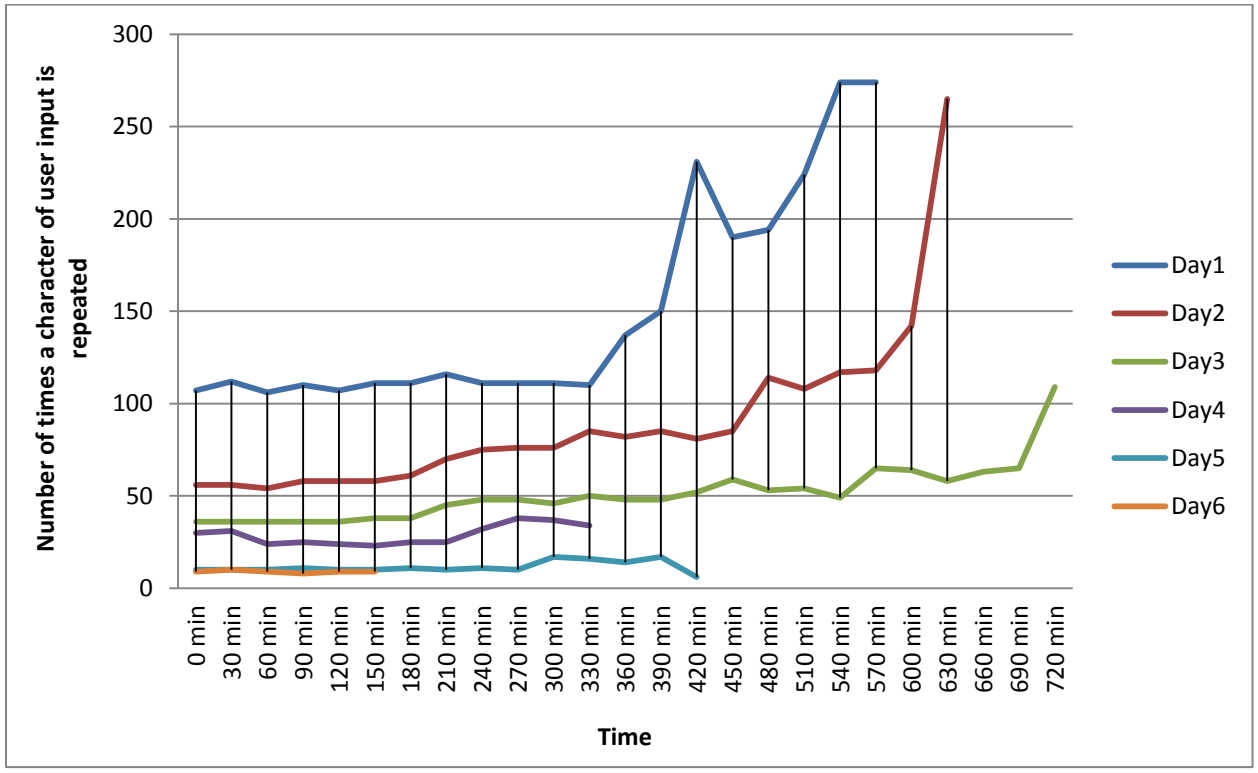


Figure 4.46 Numbers of times a character of user input is repeated

Figure 4.47 illustrates the percentage of user input found. In Day2, there are two sharp increases at 450 minutes and 570 minutes. The user input found is lower than what was found in scenarios 1 and 2. The amount of user input recovered from the allocated memory is compared with the result of previous scenario; it was discovered that large amount of user input is overwritten in the memory when the application is closed.

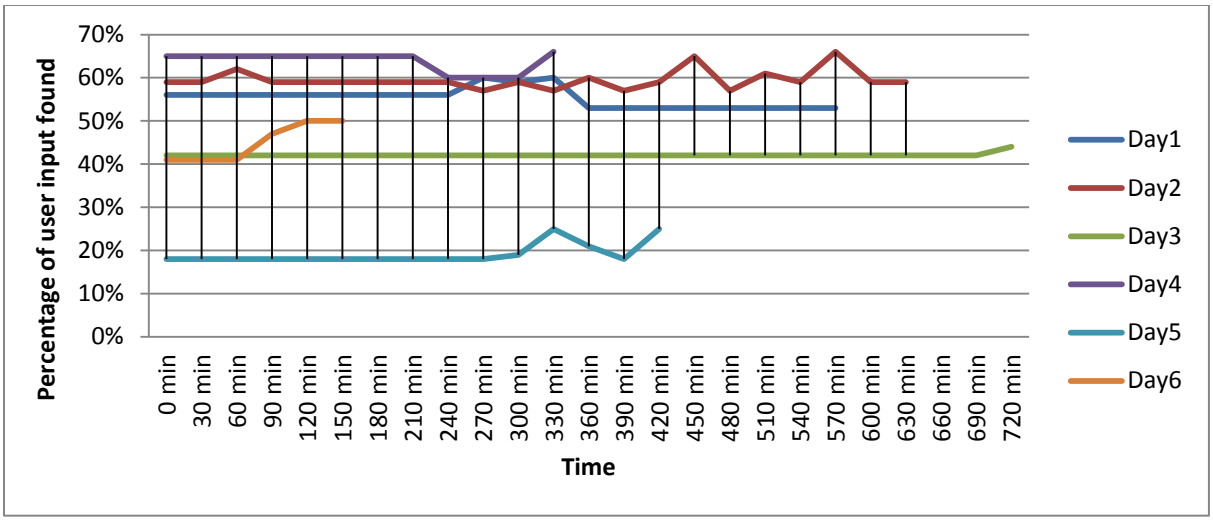


Figure 4.47 Percentage of user input found

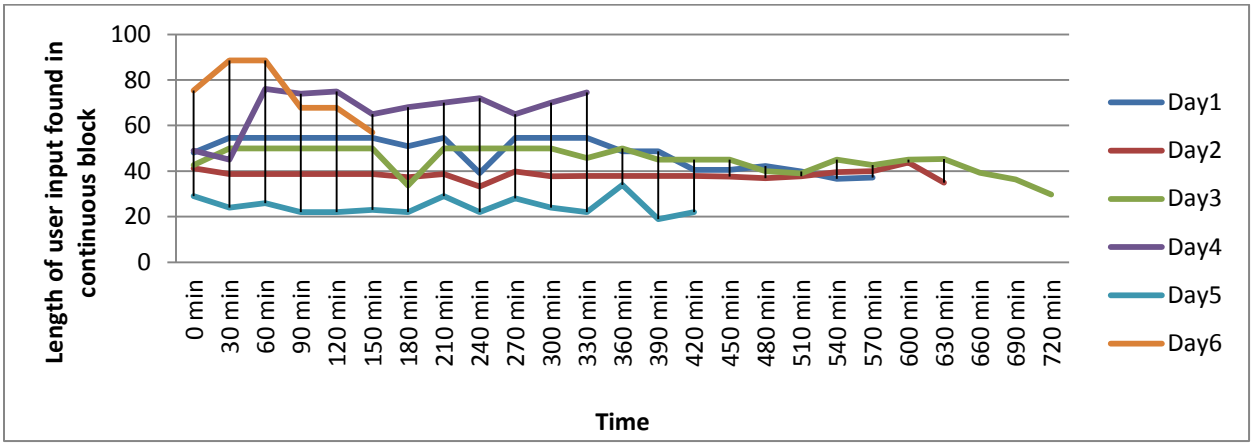


Figure 4.48 Length of user input found in continuous block

Figure 4.48 illustrates the length of user input found in continuous blocks for each day of the data capture. For example, in Day6, it was found that the user input was recorded at the highest while Day5 reported at the lowest. The two pattern techniques were used to search for user input in the application memory. When the pattern of the original user input was applied, only 48% of related data of user input was found as stored in the memory. When the pattern of some commonly used English words was applied, 30% of user input was found. The user input found was partially stored and dispersed in the memory.

4.4.4. MS Excel

In Scenario 3, time aspect of information stored by MS Excel application was investigated. Figure 4.49 shows the number of times a character of user input is repeated in the memory of the application. In Day1, the amount of user input found was at the lowest while in Day5, the amount of user input recovered was constant.

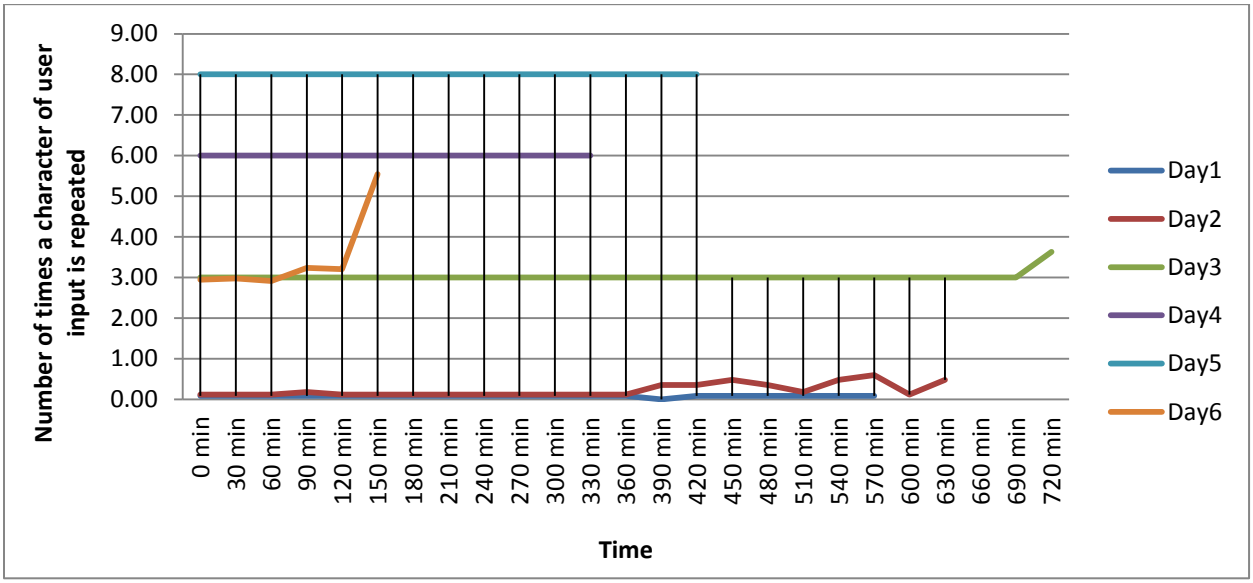


Figure 4.49 Numbers of times a character of user input is repeated

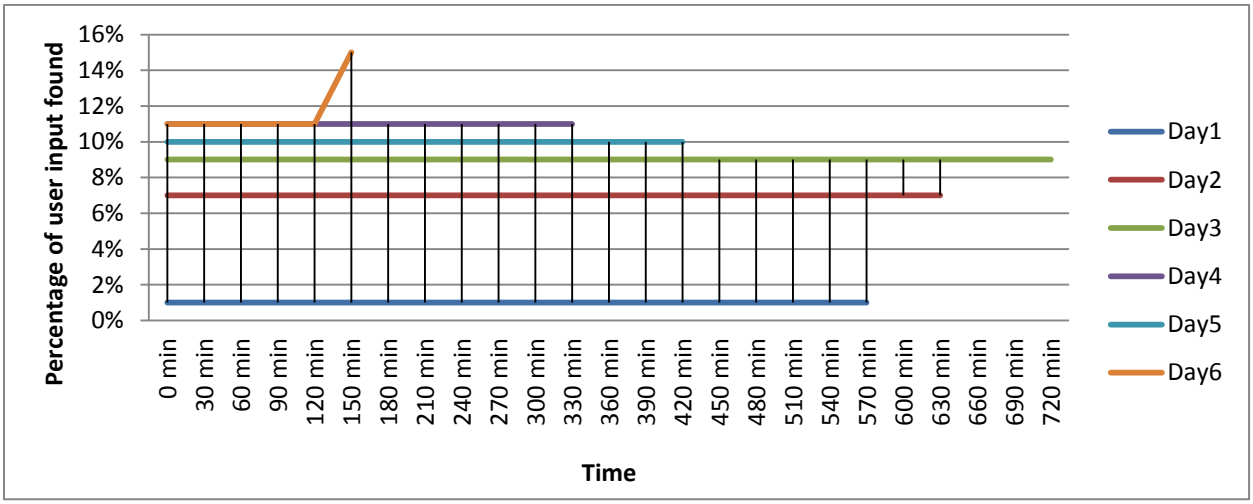


Figure 4.50 Percentage of user input found

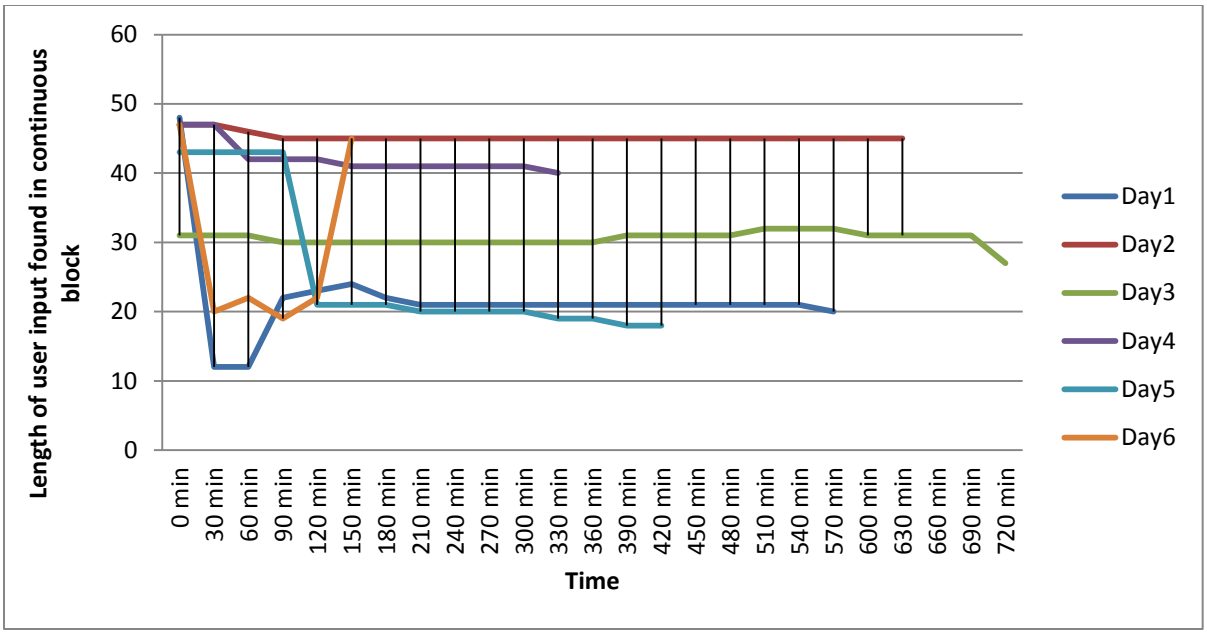


Figure 4.51 Length of user input found in continuous blocks

Recovering data proves difficult (less user input was found stored in the memory) because of the existence of numerical characters and numerical system data in the application memory (See Appendix B).

In Figure 4.50, the percentage of user input found was between 0% and 14%. Figure 4.51 shows the length of user input found in continuous blocks of the application memory. In Day1, the amount of user input found started at the highest point and dropped sharply while in Day2, the user input slightly dropped and then remain constant. When comparing the two patterns together, it was found that only 1% of related data of user input was found when pattern of some commonly used English words was used whereas, 3% of related data of user input was found when original user input of known information was applied.

4.4.5. MS Outlook

In Scenario 3, time aspect of memory allocated to user input was investigated on MS Outlook email to identify information that can only be found in the application memory. The three metrics are presented in graphs as shown below

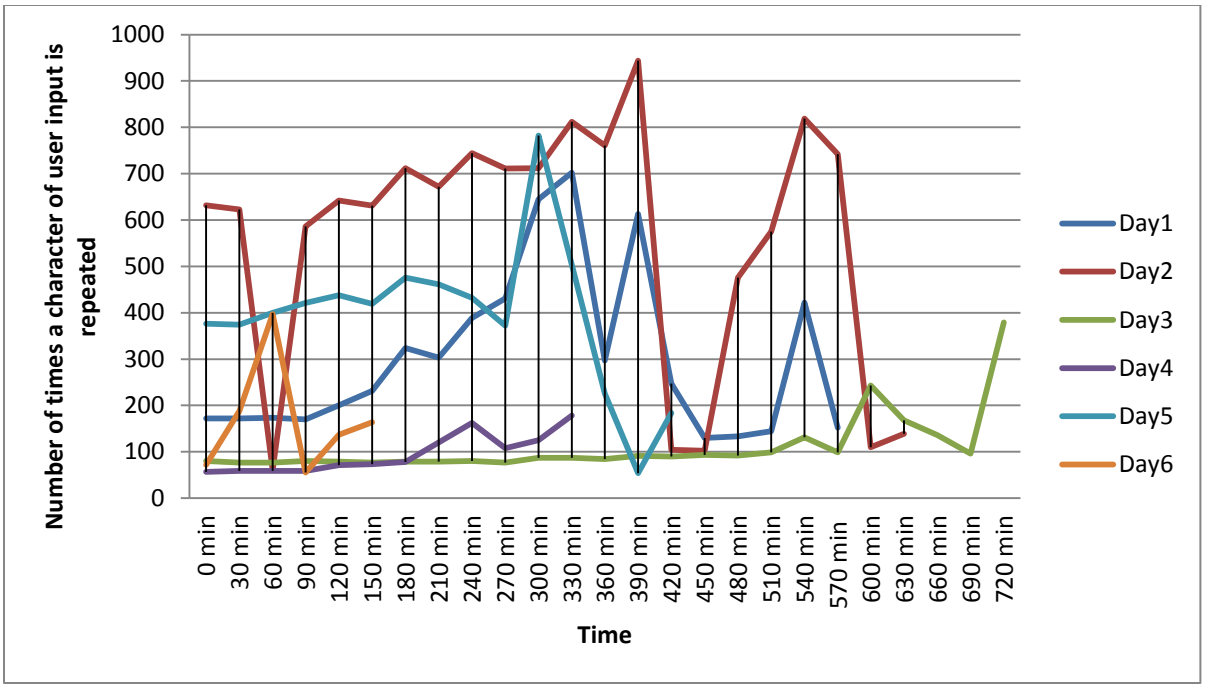


Figure 4.52 Number of times a character of user input is repeated

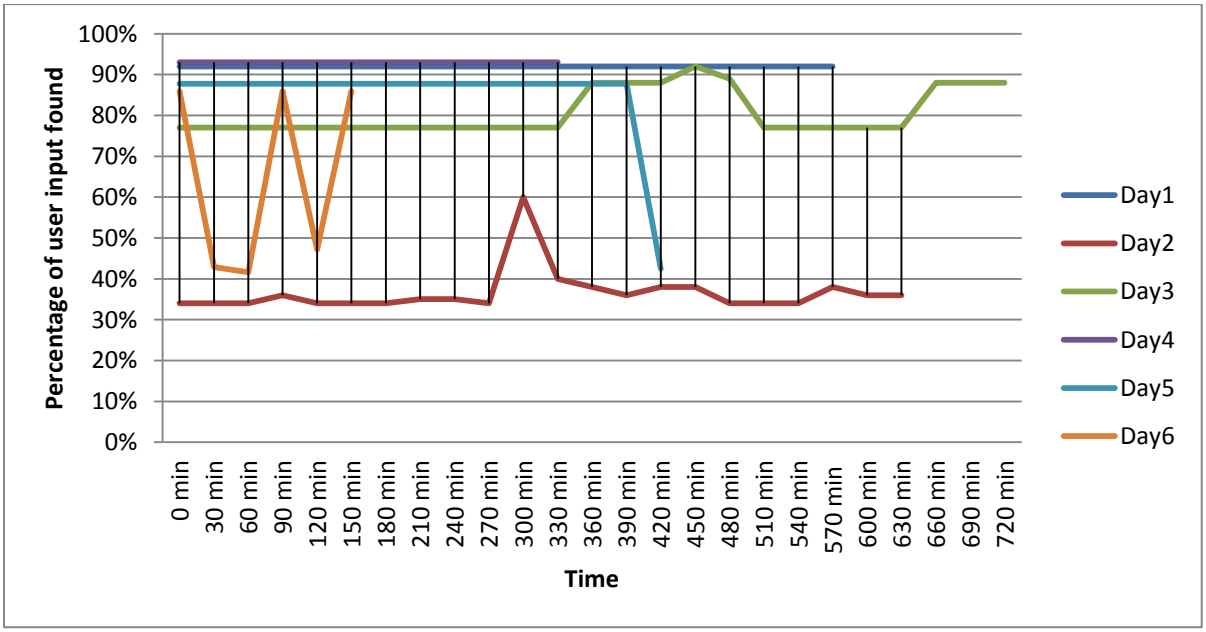


Figure 4.53 Percentage of user input found

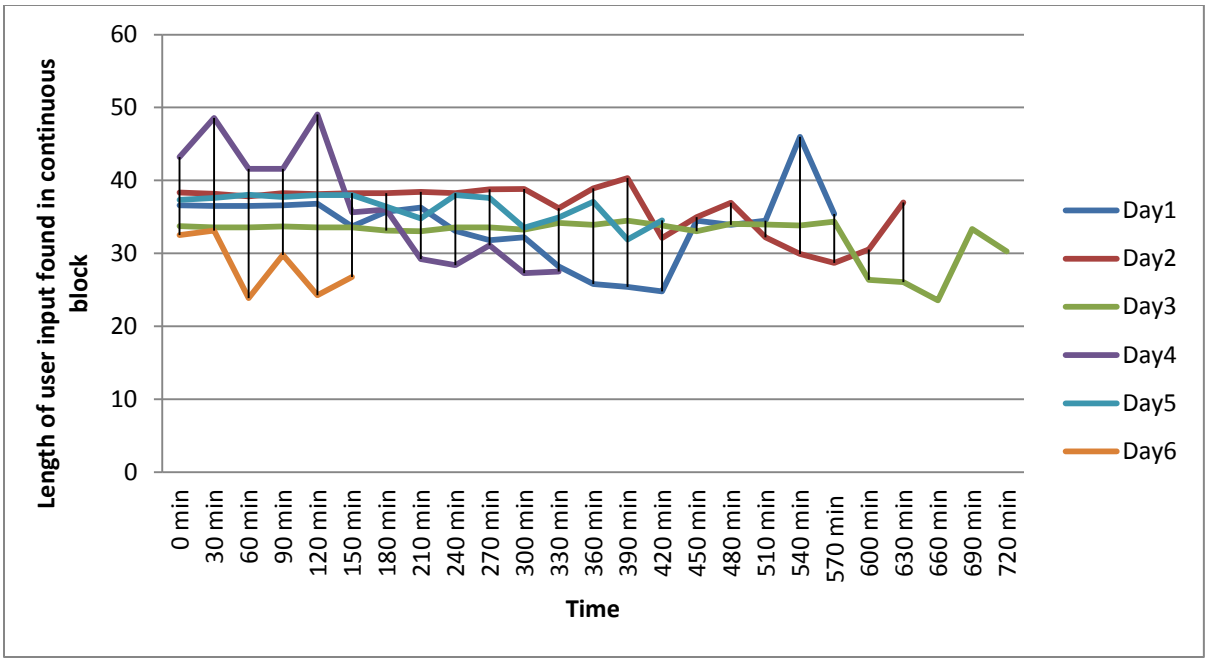


Figure 4.54 Length of user input found in continuous block

As shown in Figure 4.52, for example in Day2, the highest point was reported in 390minutes and dropped sharply in 420minutes. Data recovery is very easy because nearly all related data of user input was found in the memory allocated to this application. The emails sent and received are retained for longer in the application memory. As shown in Figure 4.53, for example in Day4 and in Day1, there is the highest percentage of user input found. The user input was retained for quite some time in the memory. As shown in Figure 4.54, for example in Day6, the user input was started as the lowest and increased slightly before dropping at 60minutes. The pattern searching technique of commonly used English words was applied with 50% of user input found in the memory allocated to this application. When pattern of the original user input was used, 75% of the user input is recovered. This information was found in whole fragment of user input.

4.4.6. MS Access

In Scenario 3, the recovery of user input on MS Access was investigated to identify the amount of data stored in the memory. The user input found is presented in the figures below. As shown in Figure 4.55, for example in Day3, the amount of user input found was at the lowest. The time memory of the application reported that user input can only be retained for short period of time. This is because the application memory retained less of user input and more of the system in-built data was recovered in the application

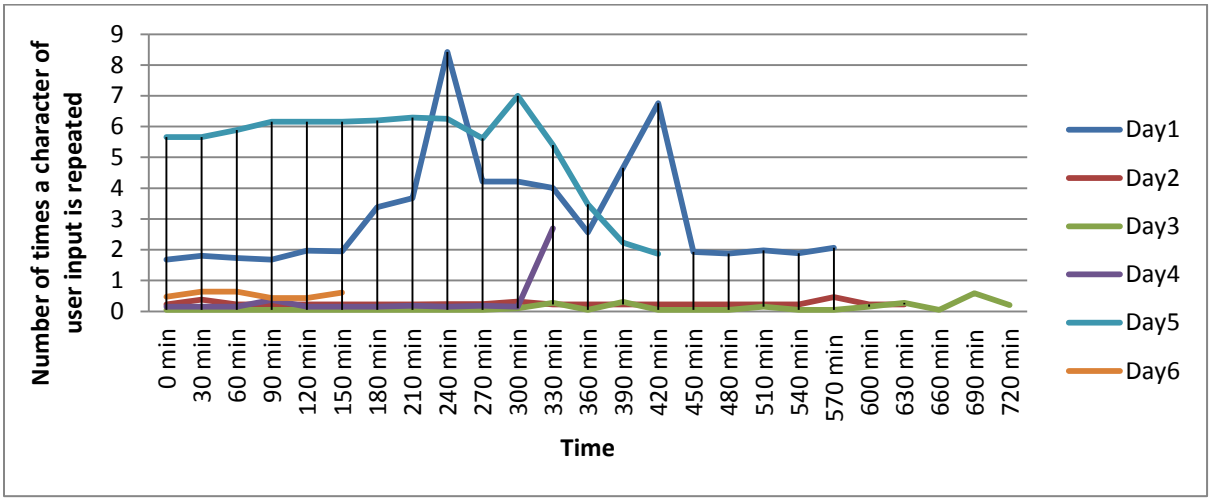


Figure 4.55 Number of times a character of user input is repeated

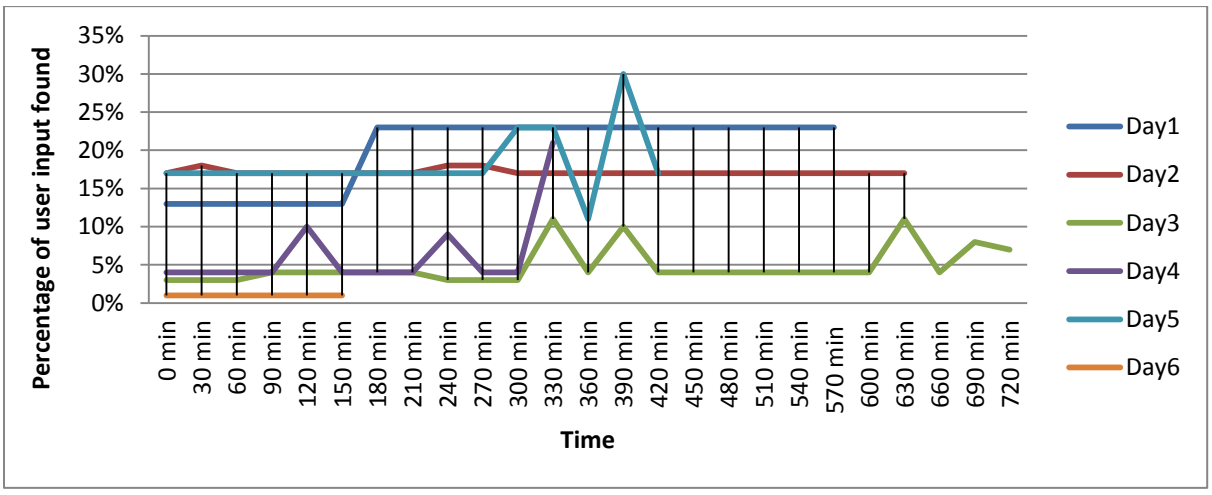


Figure 4.56 Percentage of user input found

The recovery of data is very difficult because of the in-built system defined format. The pattern matching of known information about the original user input was used and 11% of the original user input was found. When pattern matching of commonly used English words was applied, only 5% of related data was found in the memory allocated to this application. In Figure 4.56, the percentage of user input in Day5 remains at the start but, at 390minutes there is a sharp increase.

The result shows that more of the system in-built defined data lasted longer in the memory than the actual user input made (See Appendix C).

In Figure 4.57, for example, in Day3 the user input found started at the highest and dropped sharply in 30minutes of the captures and remain constant until 300minutes. It was found that user input cannot be retained in the application memory for a longer period of time. It is obvious that more system in-built defined data was recovered (See Appendix C).

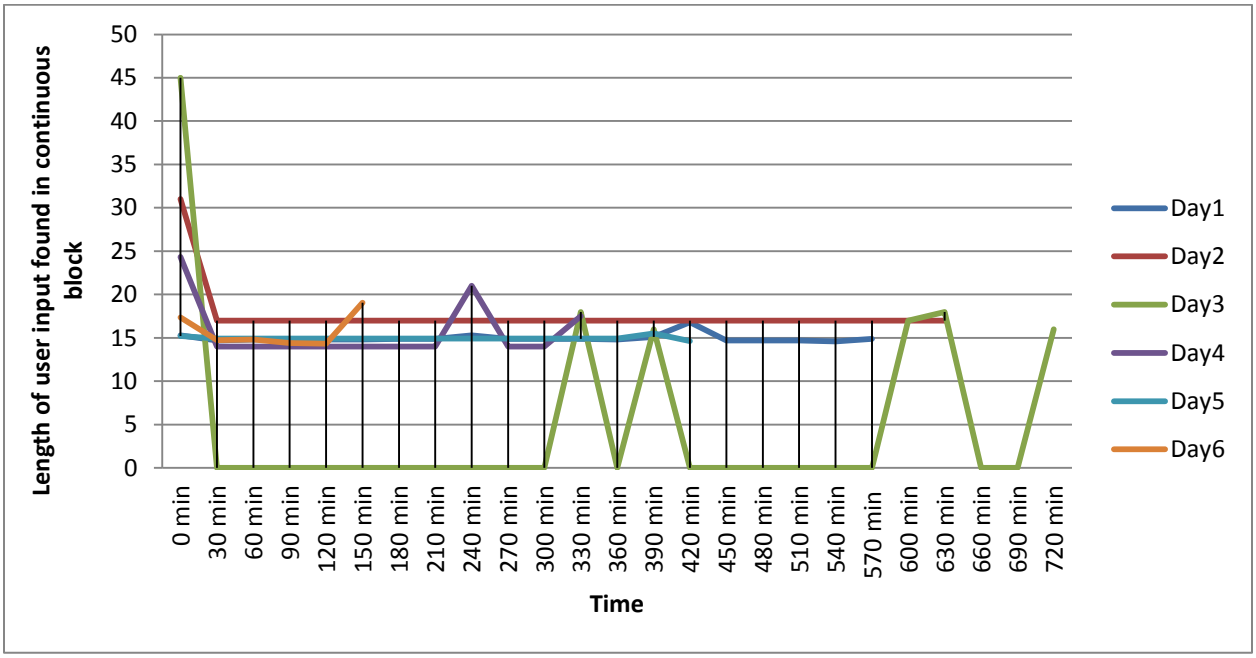


Figure 4.57 Length of user input found in continuous block

4.4.7. MS PowerPoint

The MS PowerPoint application was investigated to find out how user input was stored over time in the memory. The recovery of data seems to be very difficult. Figure 4.58 illustrates the repeated characters of user input that were found in the application memory.

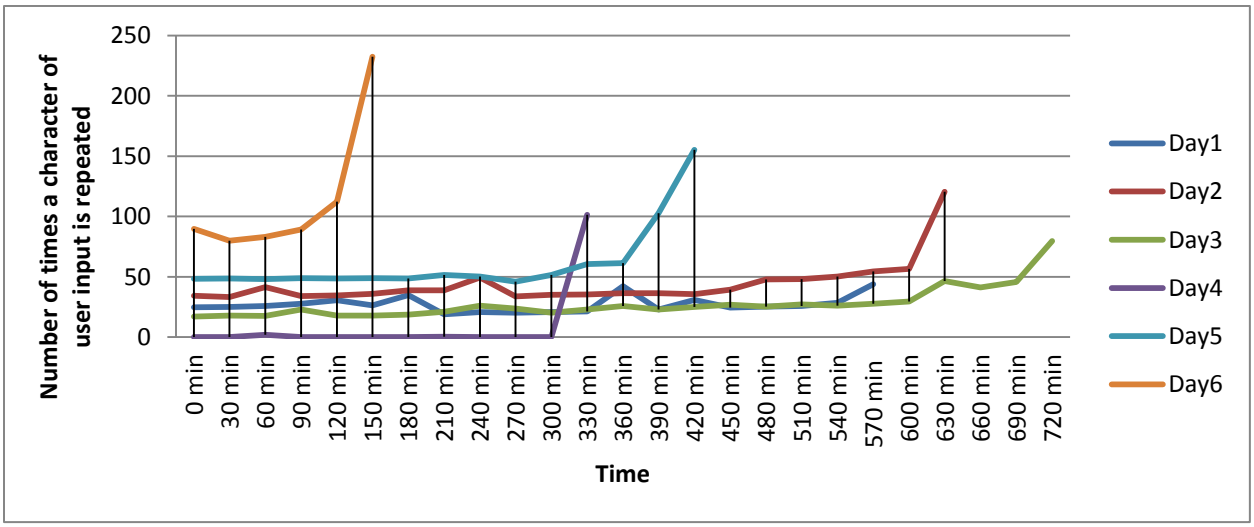


Figure 4.58 Number of times a character of user input is repeated

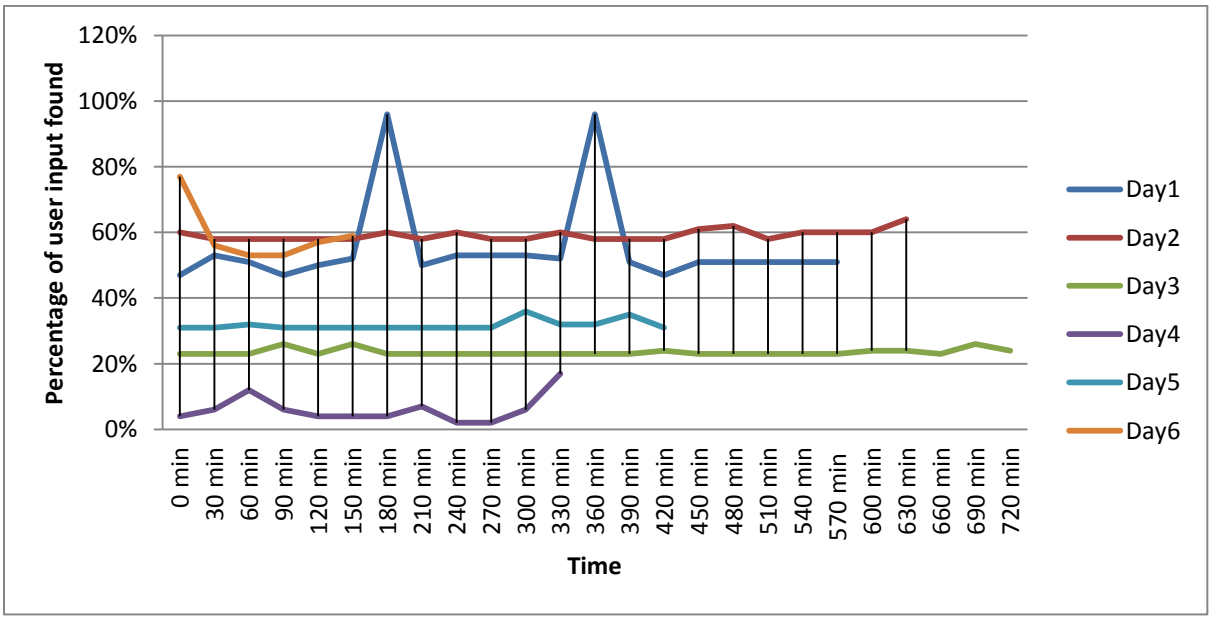


Figure 4.59 Percentage of user input found

As shown in Figure 4.59, the percentage of user input found in Day1 increases at the starting point and went up sharply at 180minutes and 360minutes. This is because the Latin character of user input reoccurred more often in the memory. The result of the user input stored in the allocated memory is less than the result found in the previous scenario. Figure 4.60 illustrates the length of user input found in continuous block.

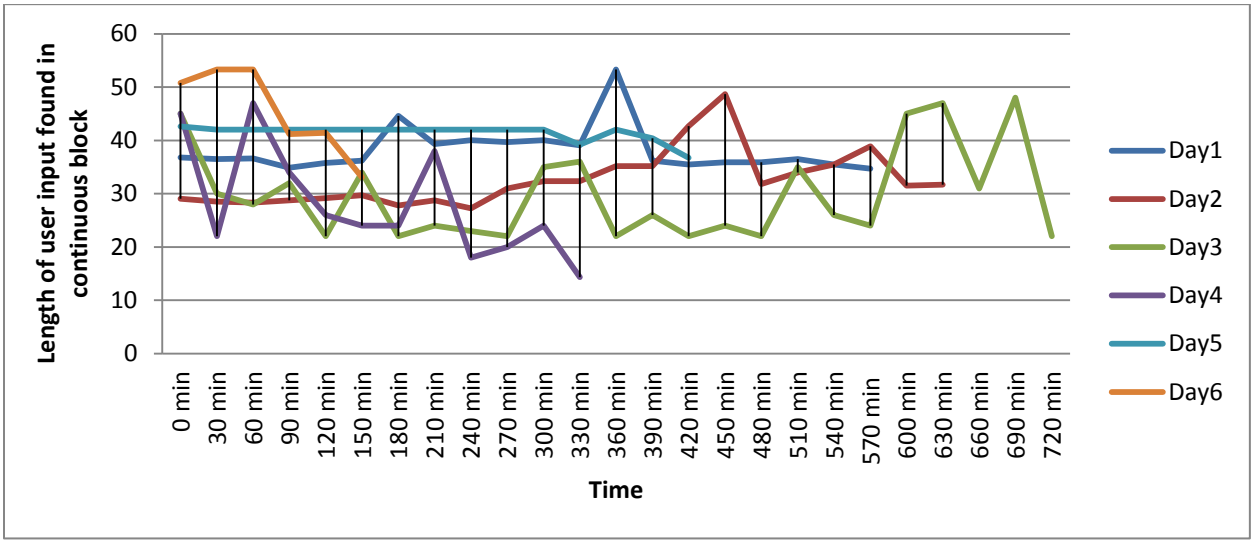


Figure 4.60 Length of user input found in continuous block

The pattern searching technique of when known information about user input was applied and it was revealed that 39% of related data of user input are found stored in the memory but, when the pattern of some commonly used English words was used, 30% of user input are found. It could be said that user input is quickly overwritten when the application has been closed.

4.4.8. MS Internet Explorer 7.0

In Scenario 3, MS Internet Explorer 7.0 was investigated to identify the user input. The recovery of data is found to be easy. As shown in Figure 4.61, for example, in Day5, the user input found was constant while in Day3, the user input found was increases in 720minutes. The highest amount of user input was found in 630minutes in Day2. As shown in Figure 4.62, the percentage of user input found is between 50% and 100%. It can be said that user input lasted for a long period of time in the memory.

The highest percentage of data was found in Day1, but this is not the case in the number of times a character of user input is repeated. It can be said that search algorithms were successful at finding data because user input is stored more often in continuous blocks of the application memory. There is strong positive correlation between the length of user input and the length of user input found in continuous block of the application memory. The amount of user input stored over time in the memory is slightly decreased in this scenario as compared to the previous one.

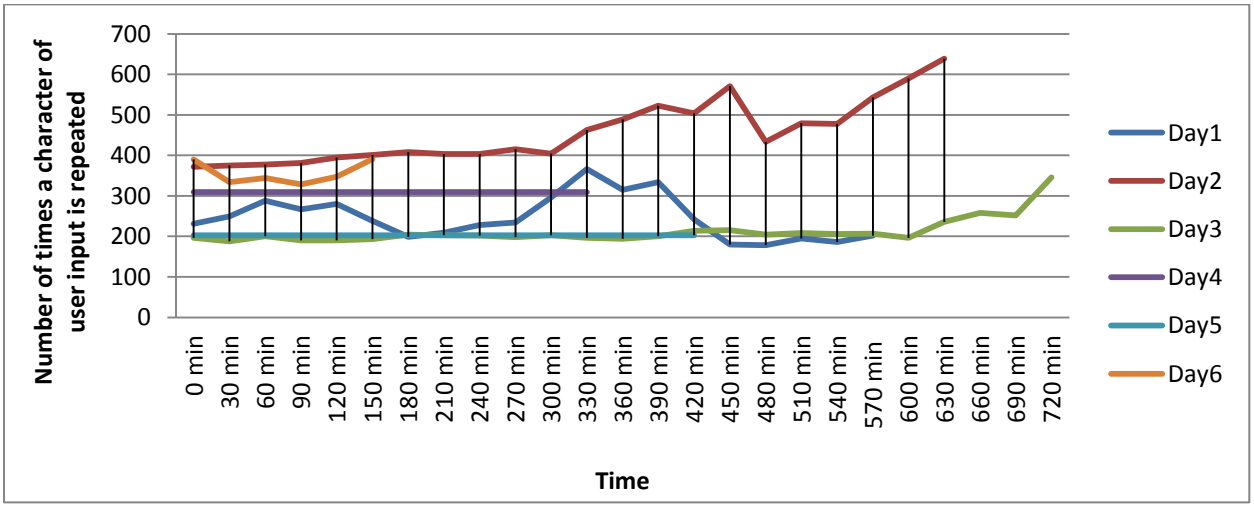


Figure 4.61 Number of times a character of user input is repeated

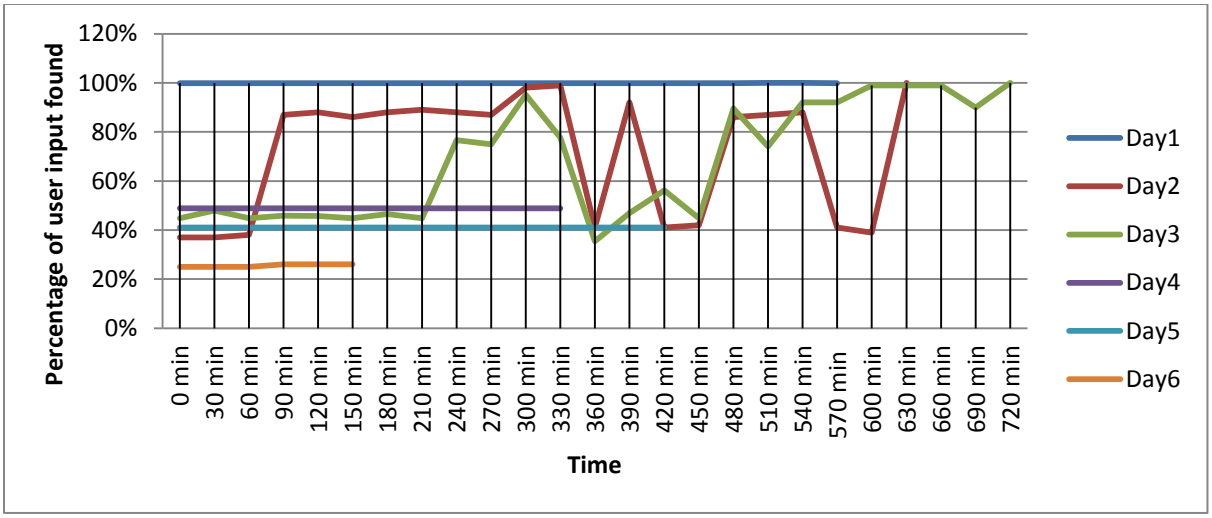


Figure 4.62 Percentage of user input found

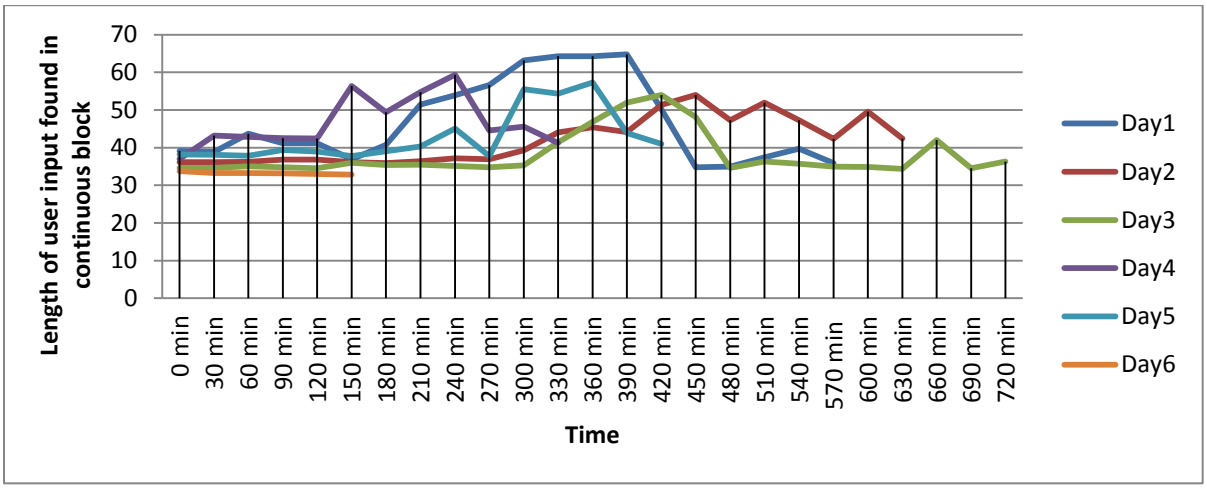


Figure 4.63 Length of user input found in continuous block

The two pattern searching techniques were used for finding data in the application memory. When the pattern of known information about user input was applied, 59% of user input was found stored in the memory, whereas only 40% of related data of user input was found stored in the application memory when the pattern of commonly used English words was applied. In this experiment, it can be said user input was partially stored and dispersed in the memory.

4.5. Scenario 4

This scenario focused on how much data is lost if the application is closed and the system is used to run other applications. At the beginning of each day, user input was made once on the applications and then the applications were closed. However, in this scenario, the computer system was used to run other applications. 100 measurements of volatile memory were captured for days at set interval of 30minutes on the application memory.

4.5.1. Length of user input

As required in Scenario 4, Table 4.4 presents the length of user input for the applications under investigation.

Table 4.4 Length of user input

Type of Application	Day1	Day 2	Day 3	Day 4	Day 5	Day 6
Adobe Reader 8.0	468	1120	478	1320	989	492
MS Word	338	327	294	423	473	491
MS Excel	176	282	204	233	148	135
MS Outlook Email	476	364	524	450	305	368
MS Access	287	325	623	311	297	358
MS PowerPoint	404	276	367	260	222	291
MS Internet Explorer 7.0	122	460	775	819	746	921

4.5.2. Adobe Reader 8.0

In this scenario, the recovery of user input on Adobe Reader 8.0 was investigated to identify the amount of data stored in the memory. This is shown in the below Figures. In Day2 of Figure 4.64, the user input found was at the highest in 510minutes. This shows the number of times a character of user input is repeated.

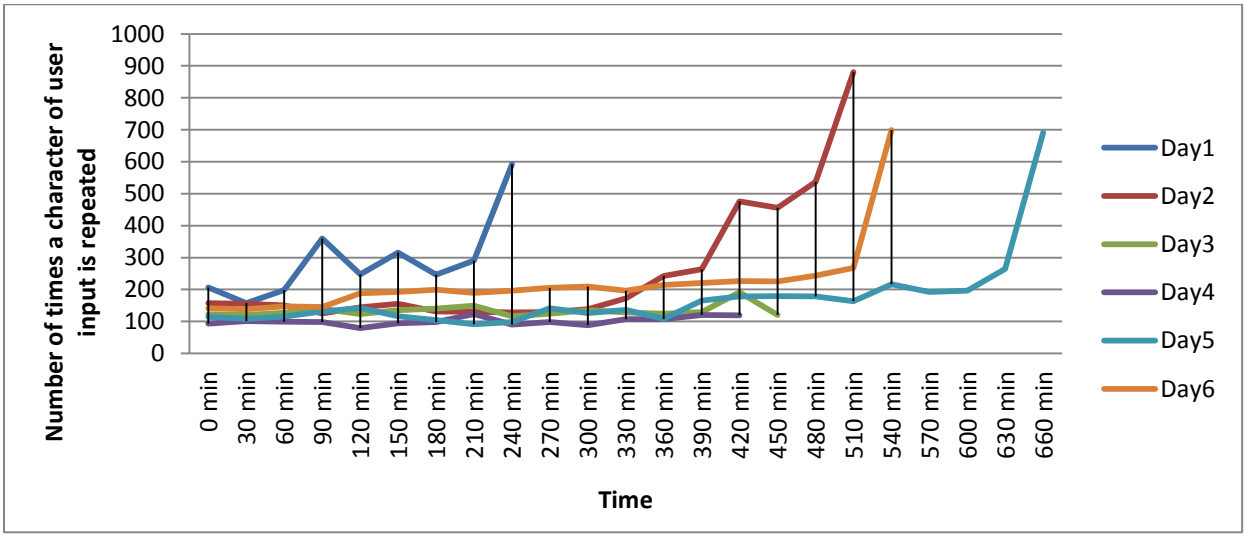


Figure 4.64 Number of times a character of user input is repeated

In Figure 4.65, the percentage of user input is between 15% and 25% in all of the days that user input was made on this application. This finding is contrary to the amount of information found in other scenarios for example, scenario 3. It is obvious that a small amount of user input like highlights and searches can appear less often in the memory when Adobe Reader 8.0 is closed. As shown in Figure 4.65, the user input cannot be retained for a long period of time.

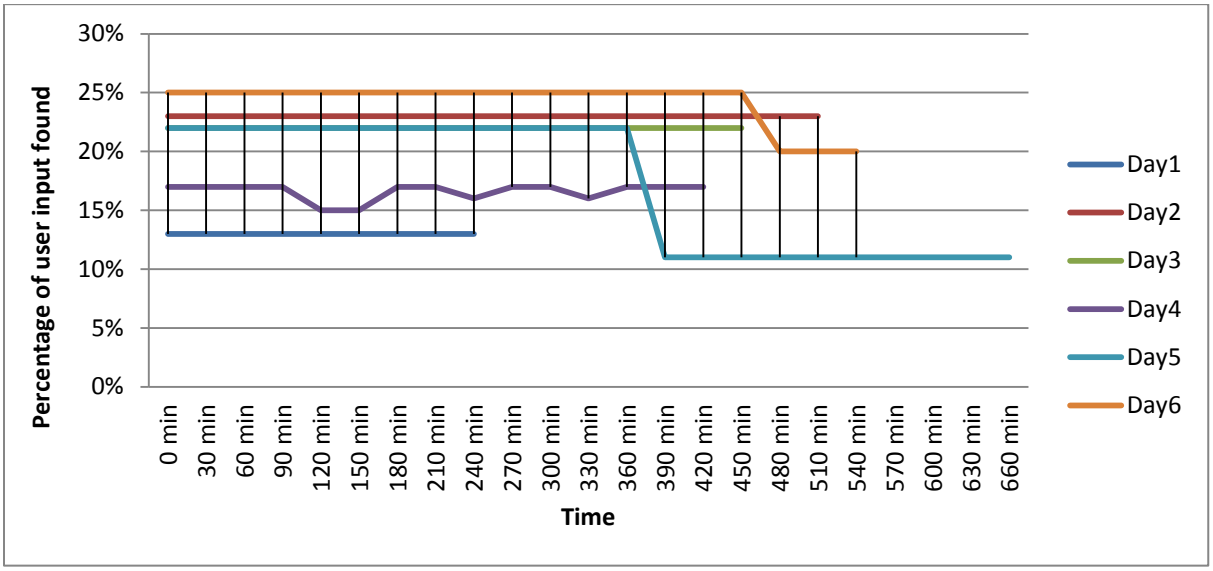


Figure 4.65 Percentage of user input found

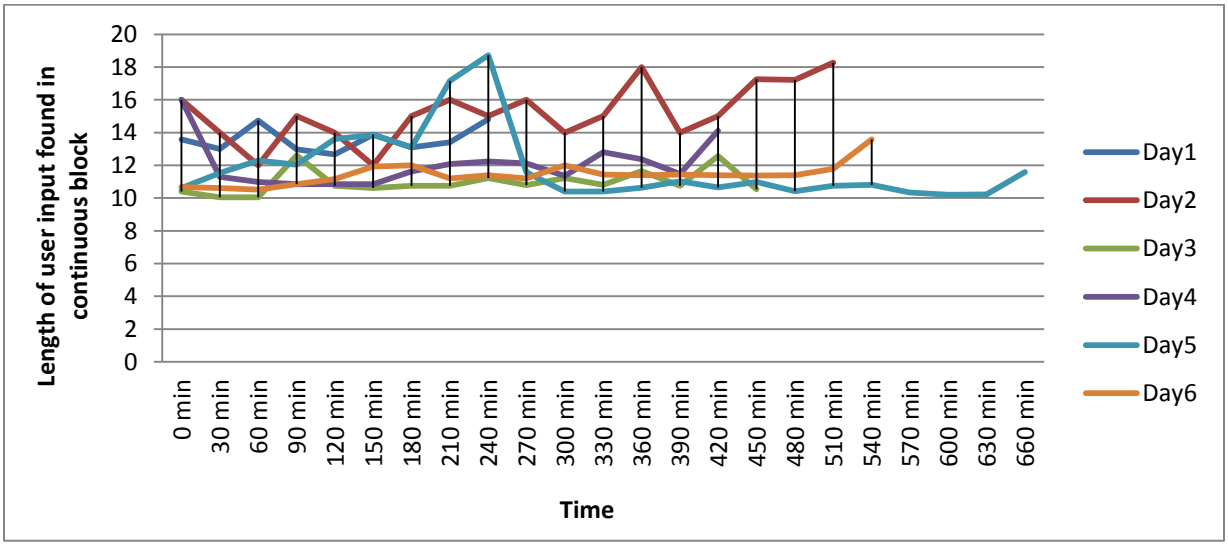


Figure 4.66 Length of user input found in continuous block

When using the pattern matching technique of commonly used English words; it was found that 20% of related data of user input was found stored in the memory whereas, when the pattern of known information about user input was used, it was resulted in 27% of the user input found. The fact remains that little of the user input was recovered when the application is closed and system is used to run other applications. Figure 4.66 illustrate the length of user input found in continuous block

4.5.3. MS Word

In this scenario, the recovery of user input on Word was investigated to identify the amount of data stored in the memory. When comparing the two patterns used, it was discovered that 26% of user input was found in the application memory when the pattern of known information about the original user input was used, whereas only 10% of user input was found when using the pattern of commonly used English words. As illustrated below, three metrics are plotted to show how much original user input was found in the memory. Figure 4.67 illustrates the character of user input found as repeated in the application memory. For example, in Day3, the user input was found at the lowest while in Day6, the highest amount of repeated character of user input was found at the highest in 540minutes. The amount of user input recovered in the memory is at the lowest on this application in this scenario. The percentage of user input found in Day6 increases at the starting point, as illustrated in Figure 4.68, and decreases sharply at 540minutes.

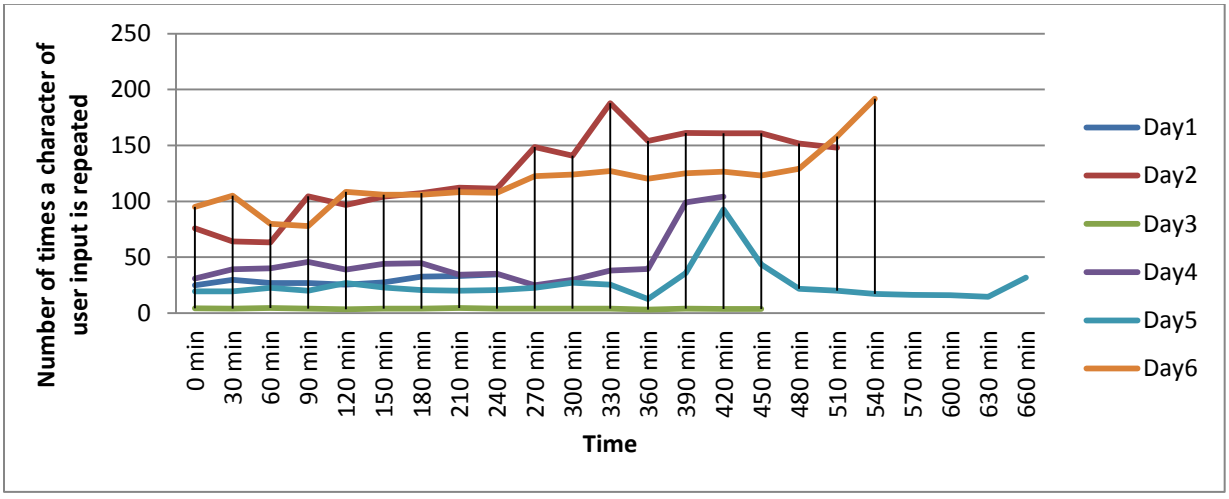


Figure 4.67 Number of times a character of user input is repeated

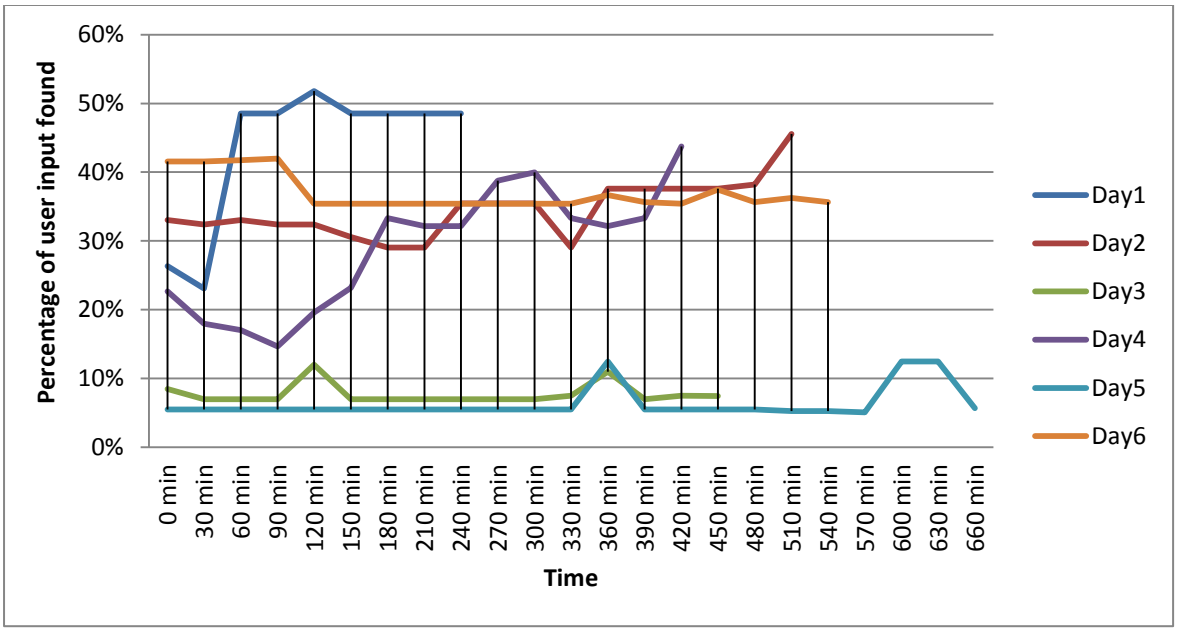


Figure 4.68 Percentage of user input found

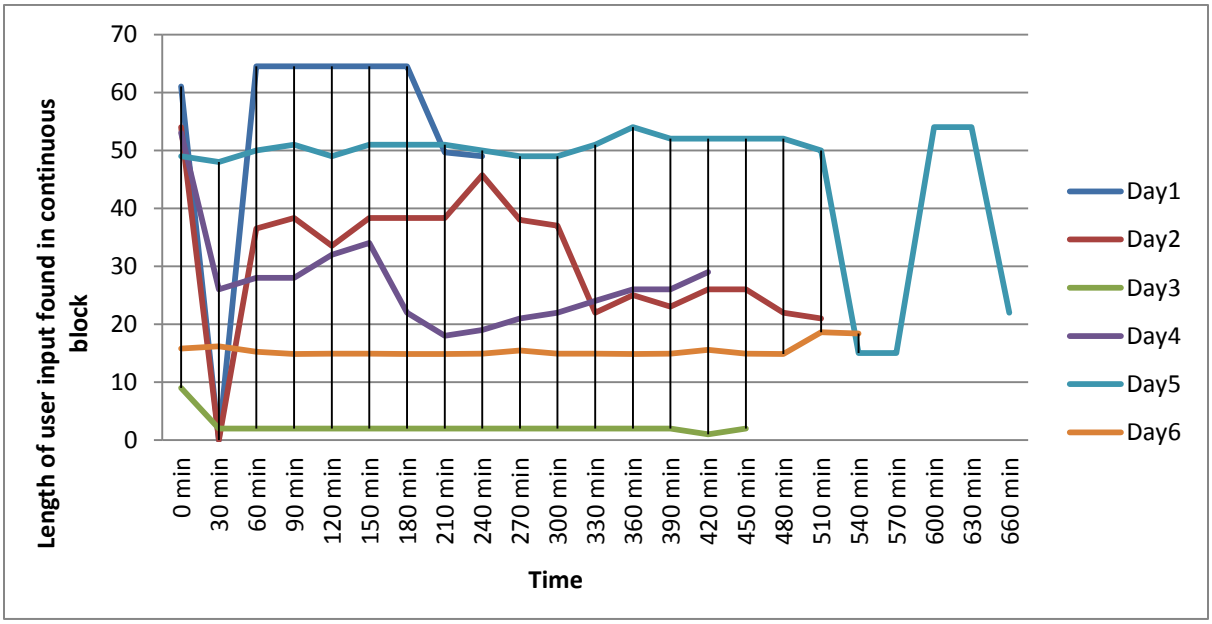


Figure 4.69 Length of user input found in continuous block

Figure 4.69 illustrates the length of user input found in continuous blocks of the application memory. Adobe uses UTF-8 format, the user input found was reduced as compared to the amount of user input found in the previous scenarios. When comparing the result found in the

previous scenarios for example, scenario 3, it was found that the amount of user input recovered is reduced. The related user input was retained for a short period of time in the memory allocated to this application. As shown in Figure 4.69, the length of user input found in continuous block, for example, in Day3, shows that the user input found was increasing at the starting point and dropped slightly in 30minutes but, remain stable until 390minutes.

4.5.4. MS Excel

In this scenario, the recovery of user input on Excel was investigated to identify the amount of user input that can be recovered from the memory. Recovering data is difficult and a large amount of user input is lost in the memory quickly when the application under investigation is closed and system is used to run other applications. Figure 4.70 illustrates the character of user input found that is repeated in the application memory. For example, in Day1, the user input found was low; data was lost easily because the system memory is used up.

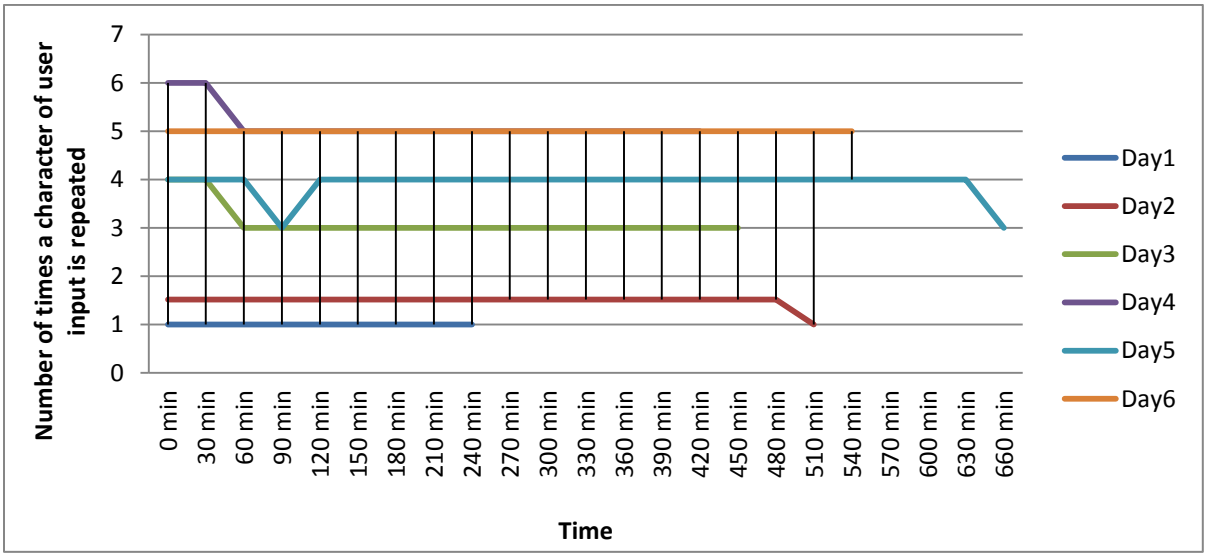


Figure 4.70 Number of times a character of user input is repeated

Figures 4.71 illustrates the amount of data recovered from the volatile memory and it can be seen that the amount of user input found is low. The percentage of user input found was at the highest in Day5 while the percentage of user input found at the lowest was reported at Day1.

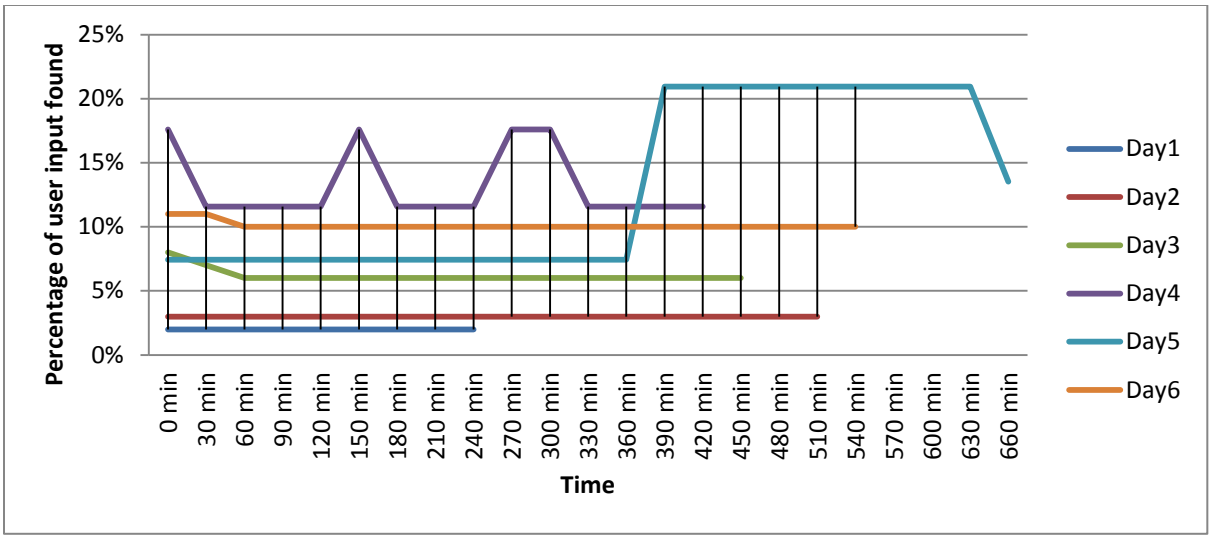


Figure 4.71 Percentage of user input found

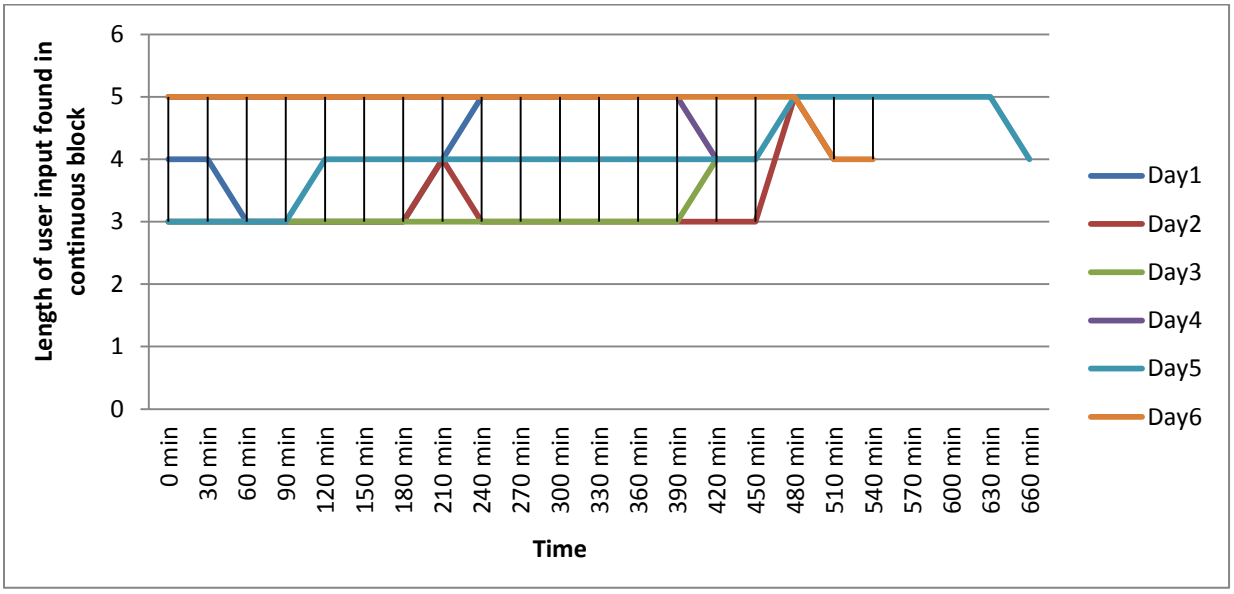


Figure 4.72 Length of user input found in continuous block

Figure 4.72 illustrates the length of user input found in continuous blocks of the application memory. When comparing the result of user input found in scenario 3, it was found that the results are different when the two patterns were applied. When the pattern of known information about user input was applied only 2% of user input was found whereas, when the pattern of commonly used English words was used, no original user input was found. Further investigation

revealed that it proved difficult to identify the original user input because of the existence of other numerical data residing in the application memory.

4.5.5. MS Outlook

In this scenario, Outlook Email was investigated to identify the information that can be found in the volatile memory. Two graphs of metrics are presented. As shown in Figure 4.73, for example, the number of times a character of user input is repeated in the application was reported. The percentage of user input found in MS Outlook in this scenario is less than what was found in scenario 3

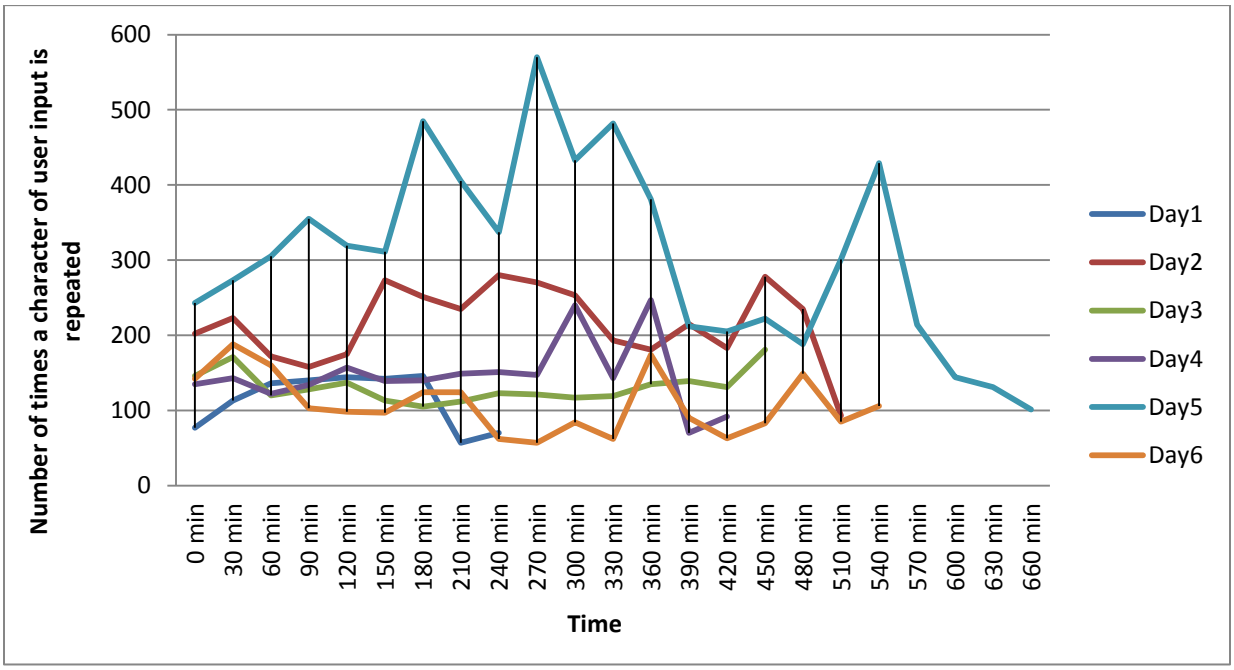


Figure 4.73 Number of times a character of user input is repeated

Figure 4.74 illustrates the percentage of user input recovered; the amount of user input found is less in this scenario than the result found in scenario 3. Although, it can be said that user input can be retained for a long period of time on this application. It can also be said that user input is stored in continuous blocks of the application memory.

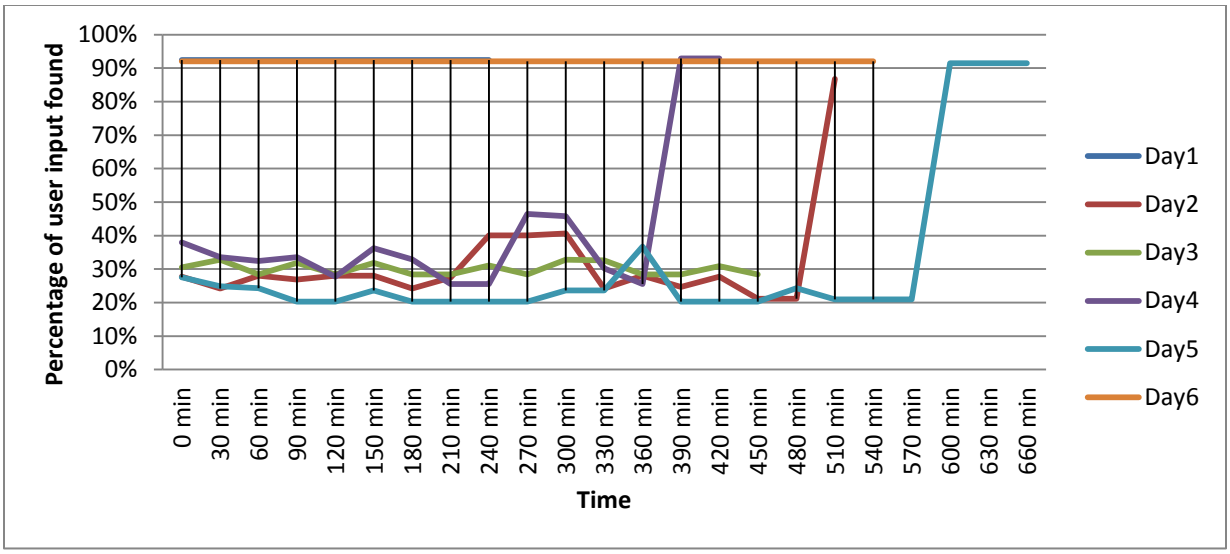


Figure 4.74 Percentage of user input found

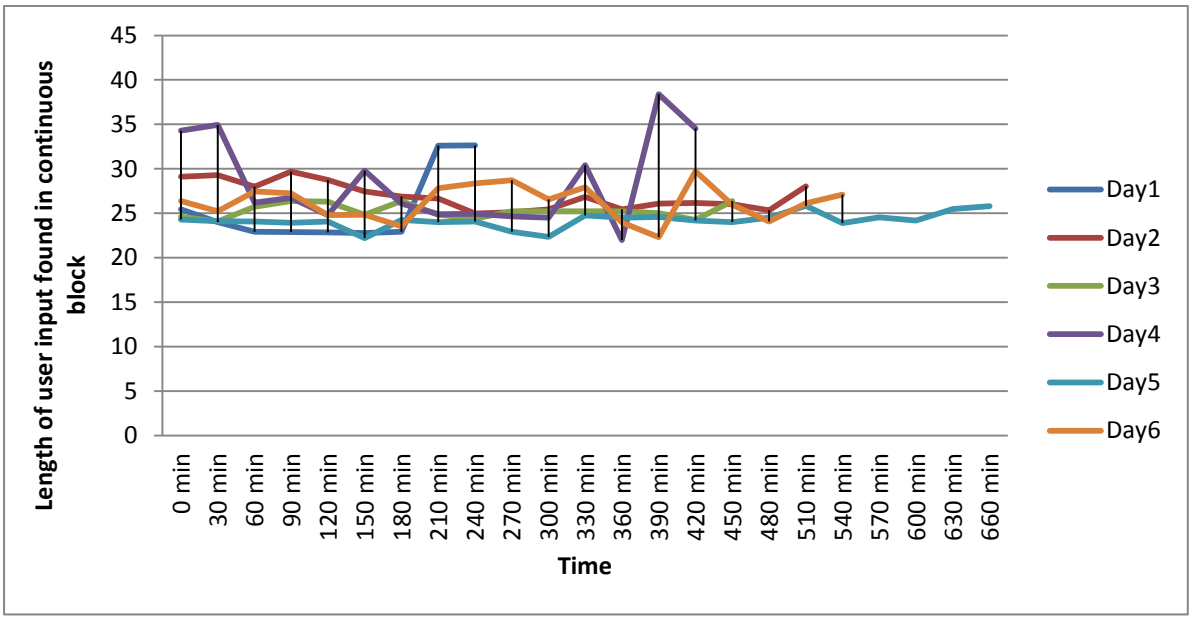


Figure 4.75 Length of user input found in continuous block

Figure 4.75, illustrates the length of user input found in continuous blocks. The pattern matching technique was used to search for user input stored in the memory allocated to this application. When the pattern of known information about user input was used, it was discovered that 53% of user input was found stored in the memory while only 30% of related data of user input was found when the pattern of commonly used English words was used.

4.5.6. MS Access

In this scenario, MS Access was investigated to identify attributes of the user input stored by the application in volatile memory. The two pattern searching techniques were applied. Figure 4.76 illustrates the number of times a character of user input is repeated in the application memory.

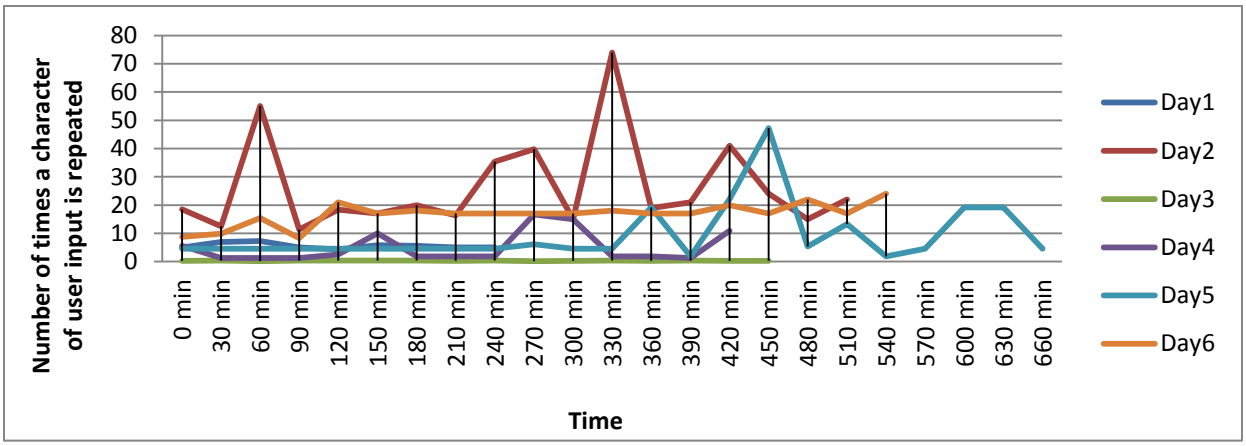


Figure 4.76 Number of times a character of user input is repeated

As shown in Figure 4.77, the percentage of user input in Day2 remains constant between times of 240minutes to 330minutes, while the highest percentage of user input was found in Day4 at 420minutes. The percentage of user input is at the lowest in Day3. It can be said that more of the in-built system defined data reoccurred more often in the memory than the original user input that was made on this application.

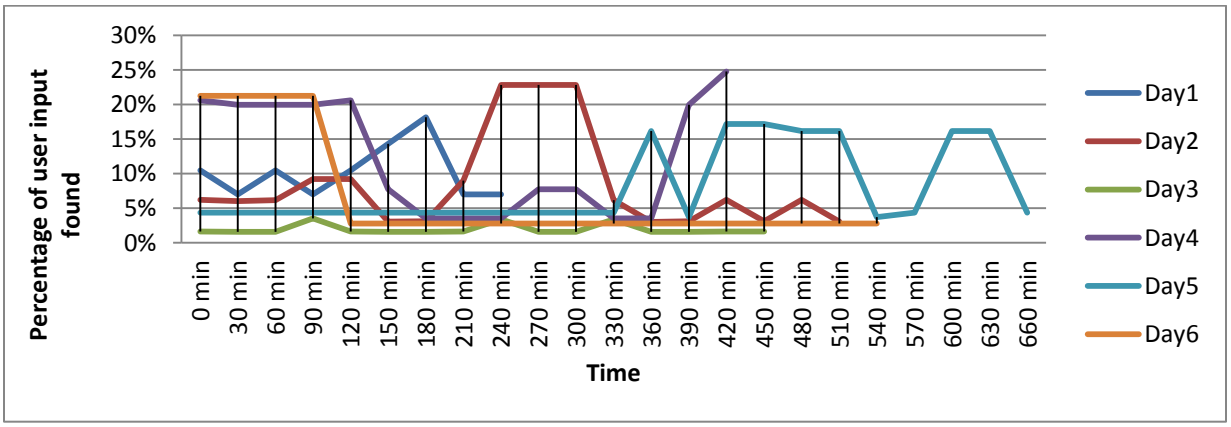


Figure 4.77 Percentage of user input found

The amount of user input in the memory of this application is less when compared with the previous scenario 3. This means that user input cannot be retained for a long period of time on this application in this scenario. Figure 4.78 illustrate the length of user input found in continuous block. When the pattern of known information about the original user input was used, it resulted in 8% of related data of user input found as stored in the memory of this application whereas, only 6% of user input was stored when the pattern of some commonly used English words was used. The user input was stored partially in the memory and it may be because of the existence of in-built system defined data that resided in the memory.

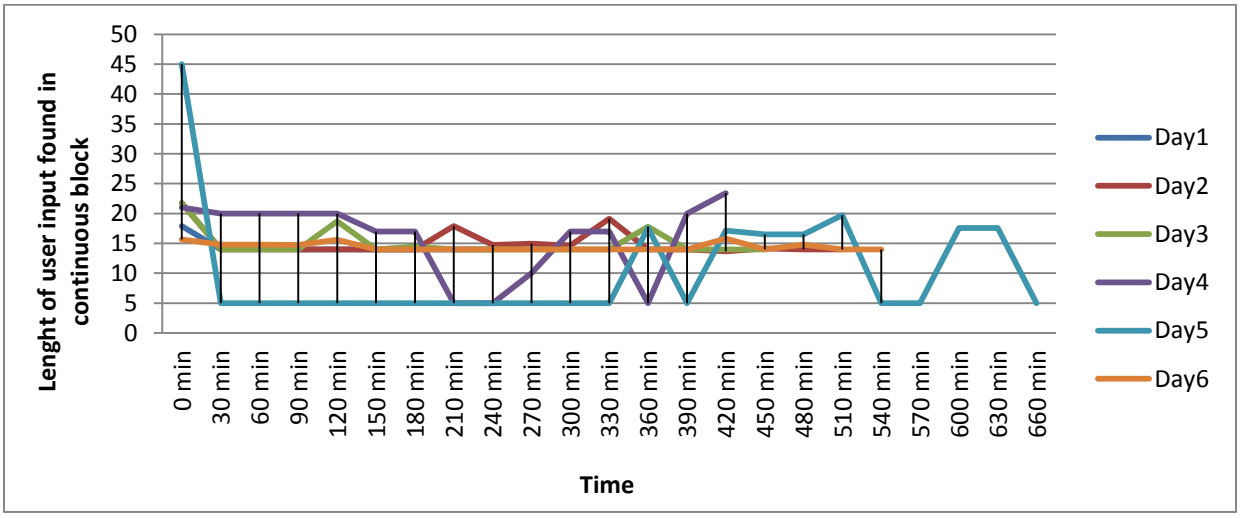


Figure 4.78 Length of user input found in continuous block

The amount of user input recovered in the memory is at the lowest on this application in this scenario. The percentage of user input found in Day6 increases at the starting point, as illustrated in Figure 4.68, and decreases sharply at 540minutes.

4.5.7. MS PowerPoint

In this scenario, the recovery of user input on PowerPoint was investigated to identify the amount of data stored in the memory. Figure 4.79 illustrates the number of times a character of user input is repeated. For example, in Day1, the user input recovered was low at the starting point, increases slightly and then dropped at 120minutes but increases sharply in 210minutes.

The recovery of data was difficult because user input cannot be retained for a long period of time in the application memory.

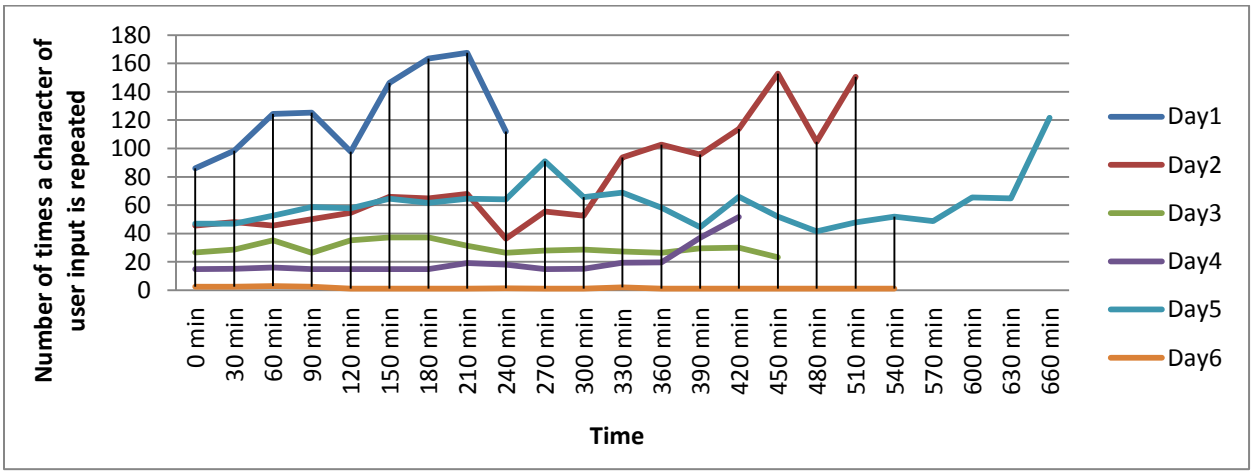


Figure 4.79 Number of times a character of user input is repeated

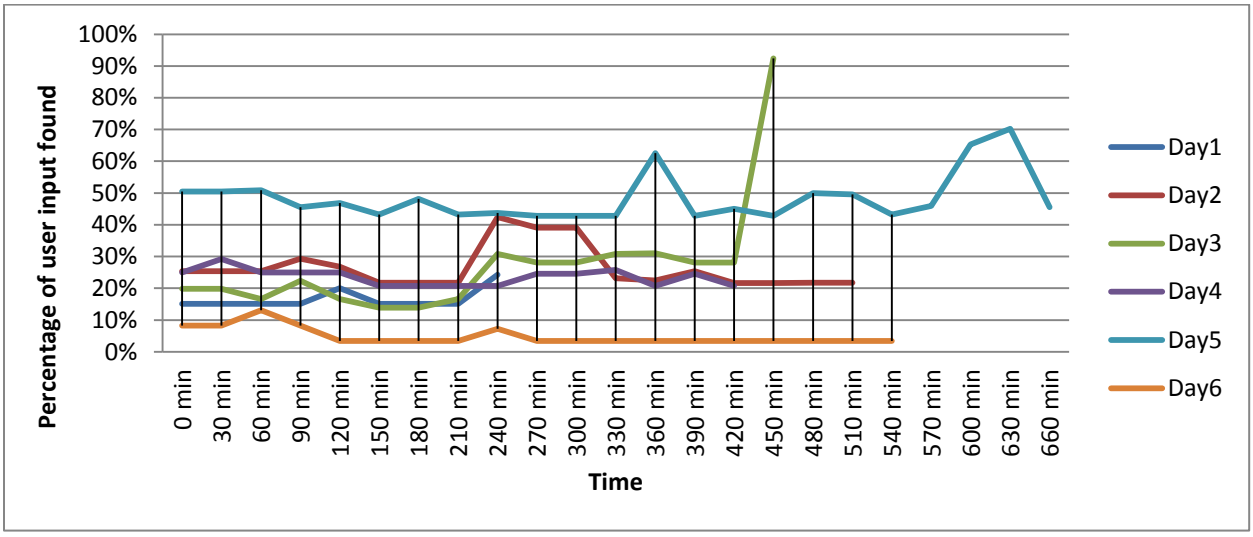


Figure 4.80 Percentage of user input found

As shown in Figure 4.80, the percentage of user input found was between 10% and 90%. A sharp increase of the user input was found in Day3 in 450minutes. The length of user input found in continuous block is shown in Figure 4.81. The search pattern of the original user input was used and it was discovered that 25% of user input was stored in the memory of the application when

pattern of known information about user input was used, whereas 11% of related data of user input was found when the pattern of some commonly used English words was used.

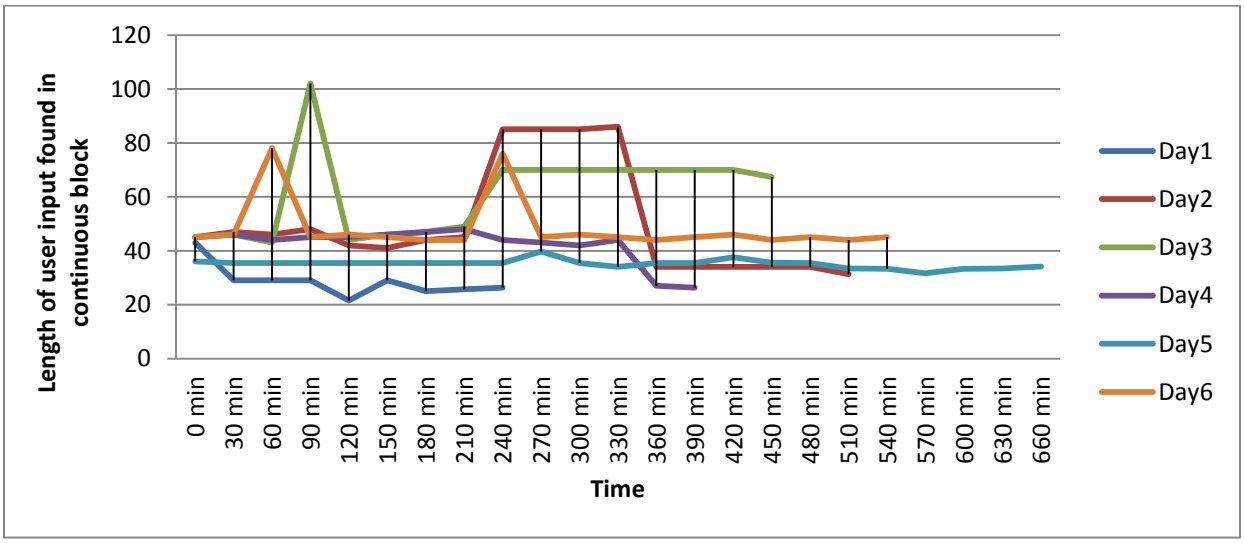


Figure 4.81 Length of user input found in continuous block

4.5.8. MS Internet Explorer 7.0

In this scenario, the recovery of user input on Internet Explorer was investigated to identify the amount of data stored in memory. This application recorded the highest amount of information found in the allocated memory in this scenario4 as compared to the amount found in the previous scenario3. Figures 4.82 illustrate the number of times a character of user input is repeated in the application memory. As shown in Figure 4.80, for example, in Day2, the highest percentage of information was found stored in the application memory. This is because user inputs like highlights and searches made on this application are retained for a long period of time in the memory. The percentage of user input found ranges between 50% and 100%. It can be said that a large amount of user input was stored over time in the application memory allocated to Internet Explorer.

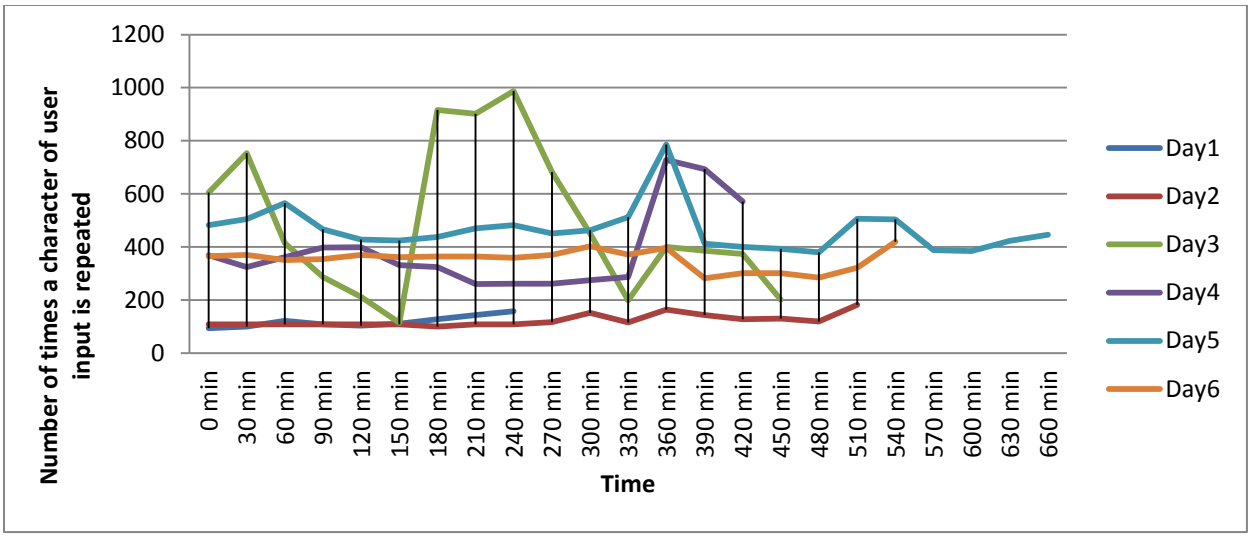


Figure 4.82 Number of times a character of user input is repeated

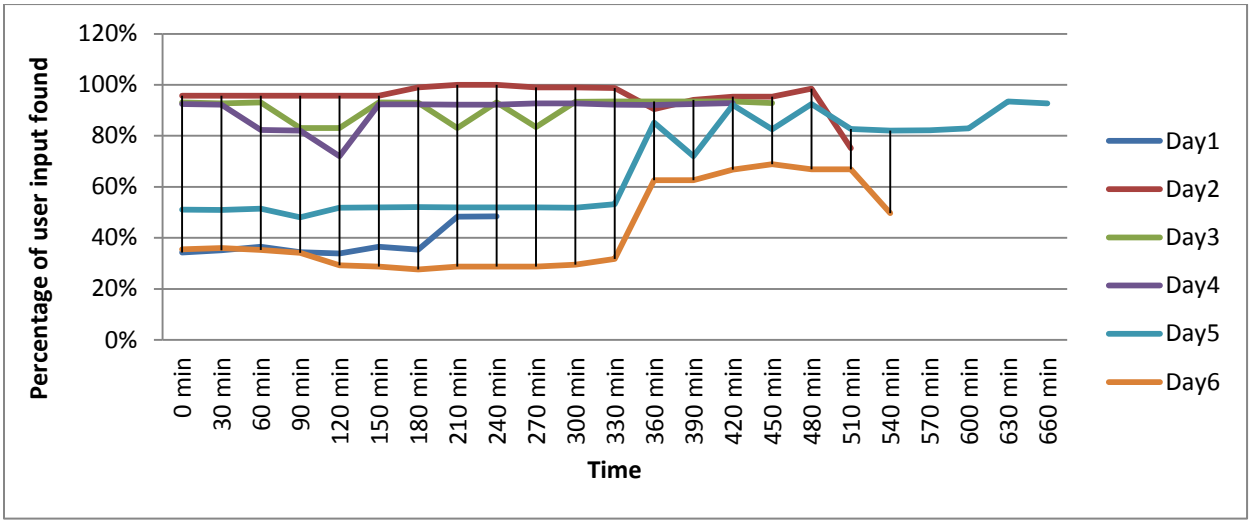


Figure 4.83 Percentage of user input found

Figure 4.83 shows the length of user input found in continuous block. Two pattern matching techniques were applied for searching for data in the application memory. When the pattern of commonly used English words was used, 37% of user input was found stored in the memory of the application, whereas when the pattern of known information about the original user input was used, 55% of related data of user input was found stored in the application memory. This information is stored in whole fragments and it was found more frequently in continuous blocks of the application memory

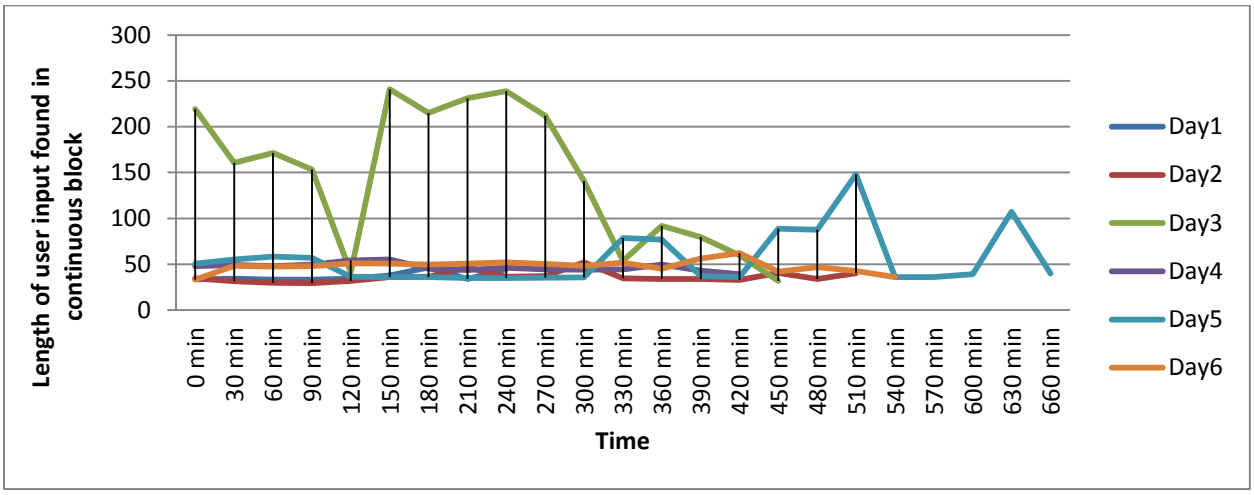


Figure 4.84 Length of user input found in continuous block

4.6. Summary

The quantitative assessment of user input is investigated to identify the user input that can be recovered and the user input stored over time in the memory allocated to an application. Following the experiments carried out and as detailed in Chapter 3, the approach of this investigation is based on the two approaches of pattern matching with known information and unknown information using commonly used English words. The percentage amount of user input found on the applications is based on the two pattern searching techniques that were described. This chapter presented the results of the quantitative assessment of application level information from the volatile memory of Windows applications. The four scenarios detailed in Chapter 3 were carried out based on the four metrics that were designed and used in the analysis of user input of the extracted application level information from volatile memory of Windows applications. The results of each of the scenarios were illustrated in this graphs as presented above. The quantitative metrics used have been discussed and assessed based on the number of times a character user input is repeated, the percentage of user input found and the length of user input found in continuous blocks of the application memory. The graphs described the amount of user input recovered and the time aspect of user input stored on the applications.

Chapter 5 Qualitative Results and Analysis

5.1. Introduction

This chapter presents the results of the qualitative assessment of the application level information that can be extracted from the volatile memory allocated to Windows applications. The four scenarios detailed in Chapter 3 were carried. The results of scenario 1 will be presented and other investigations that were carried out on scenarios 2, 3 and 4 will be described.

The qualitative assessment techniques, as detailed in Chapter 3, were used to reconstruct the user input activities on Windows applications. The reconstruction can be used to describe what the user is doing on the applications, what they have been doing and what they are using the application for. There is also a comparison between the results obtained from pattern matching with commonly used words and with the original user input. These techniques were applied on the four scenarios of the research project. The extracted user input that has been reconstructed on the applications will be presented to assess the quality of user input recovered and how the related user input is stored in the memory allocated to the applications. In scenario 1, the sample results of user input that have been extracted from the volatile memory of each of the applications are presented. Also, the time in memory of the extracted user input on scenarios 2, 3 and 4 are described. A typical example of user input that has been reconstructed on an application is presented in Appendix D. The result of the experiments indicates that the reconstruction of user input can be made using both partial and whole fragments of information. This information may be useful to forensic investigators during digital forensic investigations.

5.2. Scenario 1

In this scenario, applications were opened on the Windows system and the volatile memory was captured at set intervals of 30minutes. 100 images of user inputs were taken. The investigation focused on a specific question, *"*can all information related to how a user is using the application be recovered if the memory is captured while that application is still running?". The reconstruction of user input has been achieved for each of the applications and the sample application level information that was extracted for each of the applications are described below.

5.2.1. Adobe Reader 8.0

The extracted volatile memory of the Adobe application was reconstructed to examine what the user is using the application for. As various user input was recovered, investigations begin to identify the user input that can be presented as application level information on this application. The pattern matching was based on two patterns. There is more user input recovered when the pattern of known information about original user input is used whereas, when the pattern of some commonly used English words is used, little user input is recovered as stored in the application memory. Figure 5.1 illustrates the sample application level information that has been recovered.

```
22
C:\Documents and Settings\All Users\Start Menu\Programs\Adobe Reader
8.lnk
168
c:\program files\adobe\reader 8.0\reader\acrord32.dll
2
C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe
1
C:\Program Files\Adobe\Reader 8.0\Reader;C:\WINDOWS\system32;C:\WINDOWS
\system;C:\WINDOWS;.;C:\Program Files\Adobe\Reader 8.0\Reader\;C:
\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
12
\Documents and Settings\Administrator\My Documents\PHD TASK 2\test1-
LiveResponse.txt
13
C:\Documents and Settings\Administrator\My Documents\FUMMY-DOC
\Reconstruction\LiveResponse.pdf
1310670
Copyright
28615
Finishing
28617
executed
```

Figure 5.1 Sample Adobe Reader 8.0 application extracted

As shown in Figure 5.1, the sample application level information (A.L.I) represents what the user is typing, what user has been doing and what they are using the application for. For example, line number *"2"* identifies the process-id as *"AcroRD32.exe".* This was found repeated in the memory. The dynamic-link library *(DLL)* attached to Adobe process was found in line number *"160"* and file extension was stored as *.dll.* Also, line number *"13"* indicates what user was accessing as: *"Adobe document".* This was saved as *".pdf"* which describes the name of the

"Adobe Reader document" that user has opened. The PDF document was saved as *"LiveResponse.pdf".* This directory shows where the Adobe document was stored. This was found in line number *"1".* This information may be used to describe the type of application that the user is assessing, it can be said that user is using *"Adobe reader 8.0".* Further investigation was carried out to describe whether all related user input can be recovered. As presented in sample Figure 5.1 above, very little original user input was found. In another investigation, it was assumed that the user might be doing some highlighting or keyword searching on this application. For example, in line number "*1310670"* information is stored as *"Copyright".* It can be said that this type of information may be part of "*Adobe document that was saved as .pdf".* It may also be that user might have highlighted this word on Adobe PDF document. This information was found stored in the memory allocated to this application. Moreover, in line number *"28615",* the word *"Finishing"* was stored*;* it was assumed also that this information might be searched for by the user. It can also be part of a word of a sentence in a paragraph of "*Adobe document"* that user might have accessed. Also, line number *"28615"* the word *"executed",* this was found written in small letter. It may be that this information is part of a sentence in a paragraph of *"Adobe document"* that the user accessed. It may be that user searched for this word or it may be that user only highlighted this word in a sentence.

5.2.2. MS Word

The extracted application level information was reconstructed to form the user input that can be used to describe what the user was typing, what the user has been doing and what the user is using the application for. Figure 5.2 shows some sample application level information that was recovered from the volatile memory allocated to "Word". As shown in Figure 5.2, various user input was stored in the memory allocated to Word application and the extracted information that was found and the line numbers are recorded. This information was found dispersed in the memory. With the prospect of investigating user input activities on this application, the user input recovered can be reconstructed to form some useful statements of information. The user input found indicates what the user was typing on the application. For example, in line number *"59764",* the information found was a complete sentence of what the user was typing on the application. The word sentence consists of full stops, commas and there are no grammatical errors found.

```
5516
C:\Program Files\Microsoft Office\Office12\WINWORD.EXE
421
C:\Documents and Settings\Administrator\My Documents\PHD TASK 3\Day 1-
22042010\test1-United top world rich list despite.docx
513
United top world rich list despite.docx
1586
test1-United top world rich list despite - Microsoft Word
7116
C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\MY DOCUMENTS\PHD TASK 3\DAY 1-
22042010\TEST1-UNITED TOP WORLD RICH LIST DESPITE.DOCX
59770
Uni Port
59771
Fummy Prolite Computer
5534
HP Deskjet 6940 series
5823
C:\Program Files\Microsoft Office\Office12\WINWORD.EXE
5513
Windows NT x86
2267137
c:\Documents and Settings\All Users\Start Menu\Programs\Microsoft
Office\Microsoft Office Tools\Microsoft Office 2007 Language
Settings.lnk
2275936
Microsoft Office Word 2007.lnk
132213
Word is saving test1-United top world rich list despite:  (100%, Word
is preparing to background save test1-United top world rich list
despite:
44693
United top world rich list despite.docx
59764
MANCHESTER United have been valued as the biggest football club in the
world. The Old Trafford side, who are more than 700million in debt,
held on top spot in Forbes Magazine's list of the world's 20 most
valuable football teams. Six of that top 20 are from England - despite
the Premier League being the most indebted in Europe, according to
governing body Uefa.
77805
s 20 most valuable football teams. Six of </w:t></w:r><w:proofErr
w:type="gramStart"/><w:r><w:rPr><w:lang w:val="en-GB"/></w:rPr><w:t>
that top 20 are</w:t></w:r><w:proofErr w:type="gramEnd"/><w:r><w:rPr>
<w:lang w:val="en-GB"/></w:rPr><w:t xml:space="preserve"> from England
```

Figure 5.2 Sample MS Word application extracted

In line number *"77805"*, the information found is partially repeated in the memory for example, *"....s 20 most valuable football teams. Six..."* This information contains some grammatical errors with an incomplete sentence. There are two directories in which "*Word document"* was saved. In the first instance, the document was saved in capital letter as *"TEST1-UNITED TOP WORLD RICH LIST DESPITE.DOCX"* and this was stored in line number *"7116".* In the second time, the "*Word document"* was saved in small letter as this*: "test1-United top world rich list despite.docx".* This was found stored in the memory attached to line number *"421".* It was also

found that the name of the application *"process-id"* was found as *"WINWORD.EXE";* this was also stored in line number *"5823"*.

In line number *"5513",* there is information stored as: *"Windows NT x86".* This information can be used as the version of Windows OS that the Word application is running on. It can be said that a printer was attached to user computer because related information was found in line number *"5534"* which describes a printer. The information contains in line number *59771*: *"Fummy Prolite Computer"* can be used to describe the name given to the user's computer. Also, user input found in line number *"59770"* contains information *"Uni Port".* This information can be used to describe the "*location-name"* of the computer. Line number "*132213"* describes automatic background saving of "*Word application"*. The application level information contains partial and whole fragments of information.

5.2.3. MS PowerPoint

As required in Scenario 1, the PowerPoint application was investigated. The approach of the investigation was based on the two pattern matching techniques. Figure 5.3 illustrates the user input recovered from the PowerPoint application. The user input found represents the extracted application level information from the volatile memory of the PowerPoint application. This information can be reconstructed to find out what the user was typing, what the user has been doing and what the user was using the application for.

The information found was stored with the allocated line numbers. This information was found repeated as shown and dispersed in the memory allocated to this application but, some of this information was found in continuous blocks of the volatile memory. As shown, line number *"9"* indicated that the user had opened a PowerPoint "*.ppt slide".*

The directory of the application was stored in line number *"145"* like this *"C:\Documents and Setting…../slidelayout7.xml".* This information was found repeated. The name of the document that the user might be using contains information that was written in lower case letters and a directory is attached to it *"C:\Documents and Setting…../…test1-Rafa's keen to keep flag flying high for England.ppt".* This information was stored in line number *"316"*. The file extension of the application was stored for example,*".ppt".*

```
2
C:\Documents and Settings\All Users\Start Menu\Programs\Microsoft
Office\Microsoft Office PowerPoint 2007.lnk
3
PowerPoint is saving the AutoRecovery file.
4
"Office Theme" English (U.K.)
9
C:\Documents and Settings\Administrator\My Documents\PHD TASK 3\Day 1-
22042010/ppt/slideLayouts/slideLayout7.xml
145
application/vnd.openxmlformatsofficedocument.presentationml.slideLayout
8.xml
168
application/vnd.openxmlformatsofficedocument.presentationml.slideLayout
+xml
316
PowerPoint is saving C:\Documents and Settings\Administrator\My
Documents\PHD TASK 3\Day 1- 22042010\test1-Rafa
5795
RAFAEL BENITEZ says it is wrong to suggest the Premier League is no
longer the dominant force in European competition as his side attempt
to progress to the Europa League final.
317
C:\Documents and Settings\Administrator\My Documents\PHD TASK 3\Day 1-
22042010\test1-Rafa
476
 as his side attempt to progress to the Europa League final. to
```

Figure 5.3 Sample MS PowerPoint application extracted

As required in this scenario, related user input can be recovered. For example, line number *"5795"* contains information *"RAFAEL BENITEZ says it is wrong to suggest the Premier League is no longer the dominant force in European competition as his side attempt to progress to the Europa League final."* This information indicates what the user might be typing and what the user might have been doing on the application. This information is a typical example of a complete sentence with a full stop and there are no grammatical errors. This can be described as standard information of what a user might be using the application for. Also, similar information was stored in line number *"5795"*. This information describes the repetition of user input that was stored in the application memory. In this experiment and as earlier said above, the application level information is recovered and reconstructed.

5.2.4. MS Excel

The Excel application was investigated in accordance to the research experiment of scenario 1. The information recovered on Excel was reconstructed to find out what the user was typing on the application, what the user has been doing and what the user was using the application for. Figure 5.4 illustrates the extracted application level information recovered from Excel application. The extracted information was found stored in the memory allocated to the application with allocated line numbers. Some of this information was found repeated and dispersed. The qualitative assessment technique was adopted to assess the quality of the recovered user input that could be used as evidence for forensic investigators. As shown in Figure 5.4, the character strings of the application level information contain reconstructed information that can be used as evidence information of what the user is doing on this application. For example, user input in the memory was stored with a file extension *.xlsx*. This information can be used to describe the type of application the user was using.

```
8
C:\Program Files\Microsoft Office\Office12;C:\WINDOWS\system32;C:
\WINDOWS\system;C:\WINDOWS;.;C:\Program Files\Microsoft Office\Office12
\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
9
C:\Documents and Settings\All Users\Start Menu\Programs\Microsoft
Office\Microsoft Office Excel 2007.lnk
40
C:\Documents and Settings\Administrator\My Documents\PHD TASK 2\Day 1-
22042010\test1-wary benitez.xlsx
80390
)Wary Benitez refusing to write off rivals
80390
)Wary Benitez refusing to write off rivals
80391
Calculus Total
80392
Waryry
80393
Ben
80394
Officer
80395
Total
102932
20020202020
102933
3463762782
102934
37238464839
102935
les
102936
=sum(d7:d9)
102948
20240809880
102949
FooterTitle
102950
60722429641
```

Figure 5.4 Sample MS Excel application extracted

For example, line number *"40"* describes the document that was saved in Excel. This information is stored as "*test1-wary benitex.xlsx".* The worksheet of data was dispersed in the memory but there are two pieces of information found that can be used to describe the user input on the Excel worksheet. In the first instance, the numeric value shown in line number *"102932"* contain numeric data of *"20020202020"*. This information might be used to describe the user input entered by a user. Another line number *"102933"* contains a numeric value of *"3463762782"*and line number *"102934"* contains a numeric value that is stored like this: *"37238464839".* Also in line number *"102948",* information contains numeric data like *"20240809880"* and line number *"102950"* store a numeric value of *"60722429641".*

As shown in Figure 4.25 above, the numeric values are found in different line numbers do not contain the pound sign characters, *"£", or dollar sign characters"$".* It can be said that it was very difficult to know what the value is representing. It was also found that five different numeric values were found in the memory. In the second instance, there are five separate texts of information that were found. This information may be used to describe the alphabetic items associated with the five different numeric values that were found in the memory. For example, line number *"80392"* contains information like *"Waryry"* and also, line number *"80393"* stored data texts like *"Ben".* This information could be the name of an object but, it may also be used as an item description of the user input. In addition, line number *"80394"* contains the information *"Officer"* and line number *"80395"* contains *"Total".* It was discovered also that line number *"80391"* contain data text information *"Calculus Total".*

In this investigation, it can be said that there are five pieces of text entered by the user; this could be used to describe the items. This means that both numeric values and data text of information can be used to describe the application level information. It can be said that this information forms complete worksheets of Excel application and the numeric values with the associated line numbers may be computed, but it was difficult to link the five elements of data texts and those of the numeric values stored in line numbers of the memory together. Thus, the application level information found was partially related to the original user input made on the Excel application.

5.2.5. MS Outlook Email

As described in Scenario 1, application level information on Outlook Email application was extracted from the volatile memory of the Windows system. The investigation focused on user input that can be used to describe what the user was typing, what the user has been doing and what the user was using the application for. This information can be referred to as the application level information stored in the memory. Figure 5.5 shows a sample of user input that was recovered.

```
18
C:\Documents and Settings\Administrator\My Documents\PHD TASK 2\Day 1-
22042010\EMAIL-test1-United Richest Club in Europe.htm
19rator\My Documents\PHD TASK 2\Day 1- 22042010\EMAIL-test1-United
Richest Club in Europe.htm
29
C:\Documents and Settings\Administrator\My Documents\PHD TASK 2\Day 1-
22042010\EMAIL-test1-United Richest Club in Europe_files\filelist.xml
365
C:\Documents and Settings\Administrator\Local Settings\Application Data
\Microsoft\Outlook\Novell Gfunkyus@hotmail.co.uk-0000000b.pst
1834
C:\Documents and Settings\Administrator\Local Settings\Application Data
\Microsoft\Outlook\Novell Gfunkyus@hotmail.com (1)-0000000a.pst
53282
Dear Frank,-<span lang=EN-GB style='mso-ansi-language:EN-GB'> I got the
new today that Manchester United has been listed among others as the
best club in Europe. The news came amidst the economy crisis that among
other football club in Europe,
Manchester United top the world rich list. Despite their millions debt
and that the owner is not ready to sell the club,
the club has been rated high among other club in Europe.
294455
funkyus@hotmail.com
315002
Regards,
315003
Funky
54047
Funky Jones
54047
22 April 2010 12:17
315038
Uni Port
315039
Funky Prolite Computer
315004
Dear Frank,
315007
I got the new today that Manchester United has been listed among others
as the best club in Europe. The news came amidst the economy crisis
that among other football club in Europe, Manchester United
315008
hat among other football club in Europe, Manchester United top the
world rich list. Despite their millions debt and that the owner is not
ready to sell the club, the club has been rated high among other club
in Europe.
315009
Dear Frank,
315016
I got the new today that Manchester United has been listed among others
as the best club in Europe. The news came amidst the economy crisis
that among other football club in Europe, Manchester United top the
world rich list despite . Dtheir millions debt
```

Figure 5.5 Sample MS Outlook Email application extracted

The information was found as short and long sentences. For example, line number *"53282"* contains a long sentence of user input. Other information like *"-"* can be used to describe the language of the application. It can be said that user input was written in English Language. This information contains long sentences which can be used to describe what the user is typing on this application.

The user input that has been recovered also contains an incomplete sentence with short sentences of similar information based on the email sent or received. This information was found repeated in line numbers *"315007", "315008" and "315016".* This information was found in a whole fragment of user input recovered in line number *"53282".* This information can be used to describe what the user may have been doing on this application or what the user may have been using the application for.

Moreover, there are two different line numbers that were found stored in the memory; this information best describes the directory of the file document with the "*file extension .pst"* of Microsoft Outlook Email. There are two different email addresses attached to it. For example, line number "365" stored email address in a directory of *"C:\Documents\ ……\Outlook\Novell Gfunminiyius@hotmail.co.uk-0000000b.pst".*

A directory of: *C:\Documents…..\Microsoft\Outlook\NovellGfunminiyius@hotmail.com(1)-0000000a.pst"* contain different email address and this was stored in line number *"1834".* In this part, the email address was saved like this :*".....(1)-0000000a.pst".* This result shows that it may be that one of the email addresses was used to send email information to a recipient and the other one may be used as a receiver of this email sent. Also, the date-time stamp of the email was found in line number *"54047".* The line numbers *"53282", "294455", "315002", "315003" and "54047"* contain related user input of the email sent or received by the user.

5.2.6. MS Access

According to the research requirement of Scenario 1, the application level information was extracted from the volatile memory of MS Access. The approach of the investigation was based on the information that has been reconstructed to assess the quality of information that can be

used to describe what the user was typing, what the user has been doing on the application and what the user was using the application for. Figure 5.6 illustrates the extracted application level information from this application. As shown, the extracted information shows various user input made on this application. This information was reconstructed following the qualitative assessment method that was described in Chapter 3. The extracted information is partially related to what the user might be using the application for. For example, line number *"56"* indicates the directory in which the user access was as "*Microsoft Office Access 2007".* The information stored in line number *"57"* like *"English (United Kingdom)"* indicates that the application's file is written in *"English".*

```
56
C:\Documents and Settings\All Users\Start Menu\Programs\Microsoft
Office\Microsoft Office Access 2007.lnk
57
English (United Kingdom)
449
\Documents and Settings\All Users\Application Data\Sophos\Sophos Anti-
Virus\Logs\MSACCESS$1376 20100422 105855.156.log
27978
"C:\Program Files\Microsoft Office\Office12\MSACCESS.EXE" /NOSTARTUP
/SHELLSYSTEM [ShellOpenMacro "%1", 1]
1123
Title_Label
1124
Start_Date_Label
1126
Due_Date_Label
1128
Priority_Label
1129
Status_Label
1130
Ctl__Complete_Label
1131
% Complete_Label
1132
Assigned_To_Label
56317
Attachments_Label
56318
Attachments
77075
Assigned To
142791
AttachmentsShowOnlyRowSourceValues
142793
AttachmentsAllowValueListEdits
142795
AttachmentsColumnHeads
142797
AttachmentsColumnCount
```

Figure 5.6 Sample MS Access Application extracted

Also, in line number *"27978"* the information contains a directory with the name of the application process-id. This information can be used to describe what type of application the user was using. As investigated, there are system defined data embedded in the application memory. This information was recovered more in the allocated memory than the original user input. This means that little or no related data of original user input can be recovered from this application. For example, line numbers "*1123", "1124", "1126", "1128", "1129", "1130", "1131", "1132", "56317", "56318", "77075", "142791", "142793", "142795" and "142794"* store some related information that can be used to describe the *"system defined format"* of MS Access database. In addition, MS Access application allowed *"user own defined format"* in which a user can input different information. User can make changes to the information on the *"user defined format".* In the series of experiments carried out on this application, there is little related user input that was recovered and it can also be said that there are no related user input found that can be used to describe what the user was typing on this application

5.2.7. MS Internet Explorer 7.0

User input was extracted from the memory allocated to Internet Explorer 7.0. The qualitative assessment techniques were applied to identify what the user was typing on this application, what the user has been doing and what the user was using the application for. As shown in Figure 5.7, the application level information was extracted to assess the quality of the information that can be used to identify the original user input on this application. For example, line number *"58"* describes the directory in which user accessed Internet Explorer 7.0. This information was stored as "*Documents and Settings\.....\Internet Files\Content.IE7\English United Kingdom"*. The results show that the user might be using *"Internet Explorer 7".* It was also found that the content of information stored in Internet Explorer 7.0 was written in *"English language United Kingdom".* In line number *"8404"* information was stored as "*C:\Documents and Settings\.....\PHD TASK 2\Day 1- 22042010\iest1-IE70-".* This information can be used to identify the date and time of the original user input on this application. There is a lot of repeated information found, which shows that there is a various amount of user input that has been made on this application. The Webpages that the user might be browsing to have been recovered. For example, there are four specific WebPages that user accessed using Internet Explorer 7.0.

```
58
C:\Documents and Settings\Administrator\Local Settings\Temporary
Internet Files\Content.IE7\English (United Kingdom
8404
C:\Documents and Settings\Administrator\My Documents\PHD TASK 2\Day 1-
22042010\iest1-IE70-
84125
http://news.uk.msn.com/politics/general-election-2010/General Election:
an animal's race
470807
<news.uk.msn.com>  As the UK general election nears, political parties
have embarked on their familiar quest to attract
the backing of famous personalities. Tory leader David Cameron says he
is worried about not getting his point across as
he goes head-to-head with Gordon Brown and Nick Clegg. David Cameron,
Gordon Brown, and Nick Clegg are preparing for
the first TV debate that could change the course of the General
Election.
470899
<a href="http://msn.pressdisplay.com/pressdisplay/viewer.aspx><"
title="Today's newspapers">Today's newspapers</a>Quizzes from MSN UK
News. Who's backing who? Take our celebrity election quiz.
But do you know who's backing who? Test your knowledge of celebrity
supporters by taking our fun quiz.
Go to the questions >> How the quiz works? Play the quiz and directly
see your score Are you smarter than the rest?
Answer the questions and compare your score with the other
participants.
254460
Player">ENGB_msnvideo_no_ad</source><pageGroup>MSVUK4</en-
gb/ENGB_pa/ENGB_pa_news/pageGroup><title>Courtesy of Fox Magazine. THE
A-TEAM follows the exciting and daring exploits of Hannibal Smith and
his colorful team of former Special Forces
soldiers who were set up for a crime they did not commit.  Going rogue,
they utilize their unique talents and eccentricities to try
and clear their names and find the true culprit.
268838
<a href="http://news.uk.msn.com/in-depth/" title="Special reports">
Nick Clegg MP says the sleaze that has infested Westminster will not go
away under either Labour or the Conservatives.The leaders of the three
main parties return to the campaign trail after the first leaders'
debate, won by Nick Clegg. Gordon Brown, David Cameron and Nick Clegg
get ready for the second televised Leaders' debate. The surge in
support for the Liberal Democrats continues with a new poll putting the
party on 30% - up 10% in a week.
```

Figure 5.7 Sample MS Internet Explorer 7.0 application extracted

In the first part, line number *"84125"* contains information that the user browses to "*http://news.uk.msn.com/politics/general-election-2010/General Election: an animal's race".* Following this was the information stored in line number *"470807".* This information is related to news on the UK general election. This information can be used to describe what the user has been doing on this application. This information reported the news story of the general election amongst the party leaders and also, TV debate that could change the course of the General Election of the three major political parties in the UK. This extracted application level information is related to what the user might be doing on this application and what the user might be using the application for.

In the second part, the user accessed the webpage*"<" title="Today's newspapers">Today's newspapersQuizzes from MSN UK News".* Line number *"470899"* contains a quiz competition on the UK News about the general election. Also, the header contains information about *"Who's backing who? Take our celebrity election quiz."* This extracted application level information is related to what the user might be doing on this application and what the user might have been using the application for.

In the third part, the user browses to "*<title>Courtesy of Fox Magazine".* The information started with *"The A-TEAM follows the exciting and daring exploits of Hannibal Smith and his colourful team of former Special Forces.....and find the true culprit."* This statement of information contains a sentence with commas and full stops and this information can be used to describe the click made on the news, as may be read by the user.

In the fourth part, the user accesses the webpage: "*<a href="http://news.uk.msn.com/in-depth/"title="Special reports">*. In line number *"268838"* the information contains special reports on *"Nick Clegg MP says the sleaze that has infested Westminster.....for the second televised Leaders' debate. The surge in support for the Liberal Democrats.....putting the party on 30% - up 10% in a week".* This information is related to what the user might have been using the application for. As discussed above, the user possibly accessed four different Webpages.

5.3. Scenarios 2, 3 and 4

As detailed in Chapter 3, the scenarios 2, 3 and 4 focused on the length of time that the user input remains in the volatile memory allocated to the applications. A series of experiments were carried out for the qualitative assessment of the user input found on these applications. This was achieved by reconstructing the user input found on the applications to give answers to what the user is typing on the applications, what the user has been doing and what the user has been using the applications for. Therefore, the user input that was extracted from the applications was reconstructed. The reconstruction of user input required an investigator to arrange the user input found into a sentence or sentences of words. It may also be rearranged into a phrase. By reconstructing the user input found on the application, an investigator will be able to physically sift through the user input found on each of the applications. As detailed in Chapter 3, the reconstruction involves the user input being found and rearranged.

In many cases in scenarios 2, 3 and 4 the quality and the ability to reconstruct the user input found was not affected by the length of time before the user input was captured on each of the applications. For example, in scenario 2, the reconstruction of user input activities on applications such as MS Word, MS PowerPoint, MS Outlook Email, MS Internet Explorer 7.0, was made successfully. Applications like MS Excel, MS Access and Adobe Reader 8.0 were not easily reconstructed because there was less user input found in memory. The user input found on these applications was not affected by the length of time. After the user input has been captured, the user input found at the early and the late stages contains quality information for reconstruction purposes.

In scenario 3, the reconstruction of user input on MS Outlook Email and MS Internet Explorer 7.0 were not affected by the length of time that the user input had elapsed between the user inputting information to the computer system and the memory capture being made. The user input recovered from memory captured soon after the time that the user interacted with the system was found to be as useful as user input recovered from memory captured some time after the user had interacted with the system.

In scenario 4, the reconstruction of user input was not affected by the length of time between the memory being captured and the user interacting with the system on MS Outlook Email and MS Internet Explorer 7.0. However, the quality of user input found was not rich enough for reconstruction activities, the user input found was less significant than the user input that was found in the application memory of Scenario 3.

It was expected that Scenarios 3 and 4 would lose quality in the user input that was extracted from the application memory and Chapter 4 of this thesis has shown that a large quantity of user input was lost. The quality of the user input found has not made reconstruction easy. There are difficulties in reconstructing some cases of user input found on the application memory. For example, in scenario 3, the reconstruction of user input on MS Excel proved to be difficult because the user input found on the application memory at the early stage had little significant information that can be related as the original user input and so it was difficult to compare with the later stage. The same problem occurs on this application in scenario 4. There are little or no related data of user input that were found on the application memory for reconstruction purposes. The reason for this are thought to be that MS Excel stores in-built numeric data that resides in the memory.

The existence of this data on MS Excel has a negative effect on the ease of reconstructing the original user input made on this application. It may be that the original user input may have been overwritten by other data when the application has been closed. In scenario 4, the user input found on the applications for the reconstruction of user activities was smaller when compared to the result of user input found in scenario 3. The reason for this are thought to be that more of user input may have been lost in the memory. This information may have been overwritten by other information that was resided in the memory when the application has been closed and the system is being used to run other applications.

In addition to the user input found in scenarios 3 and 4, applications like MS Word and MS PowerPoint contains partial fragment of user input stored in the application memory. The user input was extracted, but the reconstruction of user input activities proved to be very difficult. The user input found at the early stage and the late stage of the application memory is not rich enough to carry out the reconstruction of user input activities on these applications.

In Adobe Reader 8.0 and MS Access, less data of user input was found in the memory of these applications. It can be said that a large amount of user input was lost in the memory of these applications. The information on the quantity of data lost is detailed in Chapter 4. For example, in scenario 3 and 4, the quantity of user input lost was high for MS Word, MS PowerPoint, Adobe Reader 8.0, MS Excel and MS Access applications. The reason for data loss may be that the user input cannot be retained for a long period of time in the memory of these applications.

Another reason for data loss may also be that the original user input has been overwritten when the applications were closed and Windows system is used to run other applications. However, the quality of user input found on these other applications (MS Word, MS PowerPoint, Adobe Reader 8.0, MS Excel and MS Access) has been found to be poor. Therefore, the reconstruction of user input activities to form a sentence of words or a phrase of words was not possible.

5.4. Summary

The qualitative assessment of user input activities on the volatile memory of commonly used Windows applications was achieved. In scenarios 1, sample information about the user input found on the applications was presented. The user inputs are stored as fragments of information. The sample application level information was extracted using the two pattern matching techniques that were applied. There is more user input recovered when the pattern of known information about original user input is used whereas, when the pattern of commonly used English words are used, little of user input was recovered from the application memory. In Scenario 1, the extracted information that has been reconstructed was presented for each of the applications. The approach of the investigation details what the user was typing on the application, what the user has been doing and what the user has been using the application for. The information on each of the applications may be useful to forensic investigators during digital investigations. Scenarios 2, 3 and 4 describe the quality of user input found in the applications as extracted for reconstruction purposes. The extracted application level information may be affected by the quality and the ability to reconstruct the user input found on the applications based on Scenarios 2, 3 and 4. This process depends on the time memory aspect of user input stored on the application memory at both the early stage and the late stage of reconstructing the user input found on these applications.

Chapter 6 Proposed Framework or Model of Digital Investigation Process of Application Level Information

6.1. Introduction

This chapter described a proposed framework or model that encompasses the work that has been done on application level information and for future research work. The proposed model is based on four scenarios of the research work detailed in Chapter 3 including the results and analysis of the data presented in Chapter 4 and 5. The proposed framework describes and demonstrates how a forensic investigator can carry out digital crime investigation of user input on volatile memory of Windows applications. The steps implemented to design the four phases of digital investigation process of application level information will be discussed.

6.2. Digital investigation framework of application level information

The digital investigation process framework of application level information (DIPFALI) is based on some existing frameworks. This proposed forensic investigation framework may also be useful for future research work. Several literatures on digital forensic investigation frameworks were reviewed as detailed in Chapter 2 of Section 2.4.1. The digital investigation process framework of application level information is built upon the research works of (Brian & Eugene, 2003), and (Felix & Bastian, 2007).

The model of (Brian & Eugene, 2003) includes six major stages, preservation of digital scene, survey for digital evidence, document evidence and scene, search for digital evidence, digital crime scene reconstruction, and presentation of digital scene theory. The framework of (Felix & Bastian, 2007), focused greatly on the analysis of data collected from digital investigation and it consists of pre-incident preparation, pre-analysis, analysis and post-analysis (Felix & Bastian, 2007).

The aim of this new model is to combine the two concepts of digital framework of (Brian & Eugene, 2003)**,** and (Felix & Bastian, 2007) to improve the overall process of digital investigation on application level information.

The digital investigation process framework of application level information (DIPFALI) is a proposal for a new process model to investigate the user input stored in the volatile memory of Windows applications. In fact, the four phases of digital investigation of application level information somewhat resembles a computer forensic investigation which is embedded into the digital crime scene investigation procedure of (Brian & Eugene, 2003). Hence, the proposed digital investigation process framework of application level information (DIPFALI) was designed. This includes preservation phase, searching phase, reconstruction phase and assessment phase. Figure 6.1 details four phases of digital investigation process of application level information.

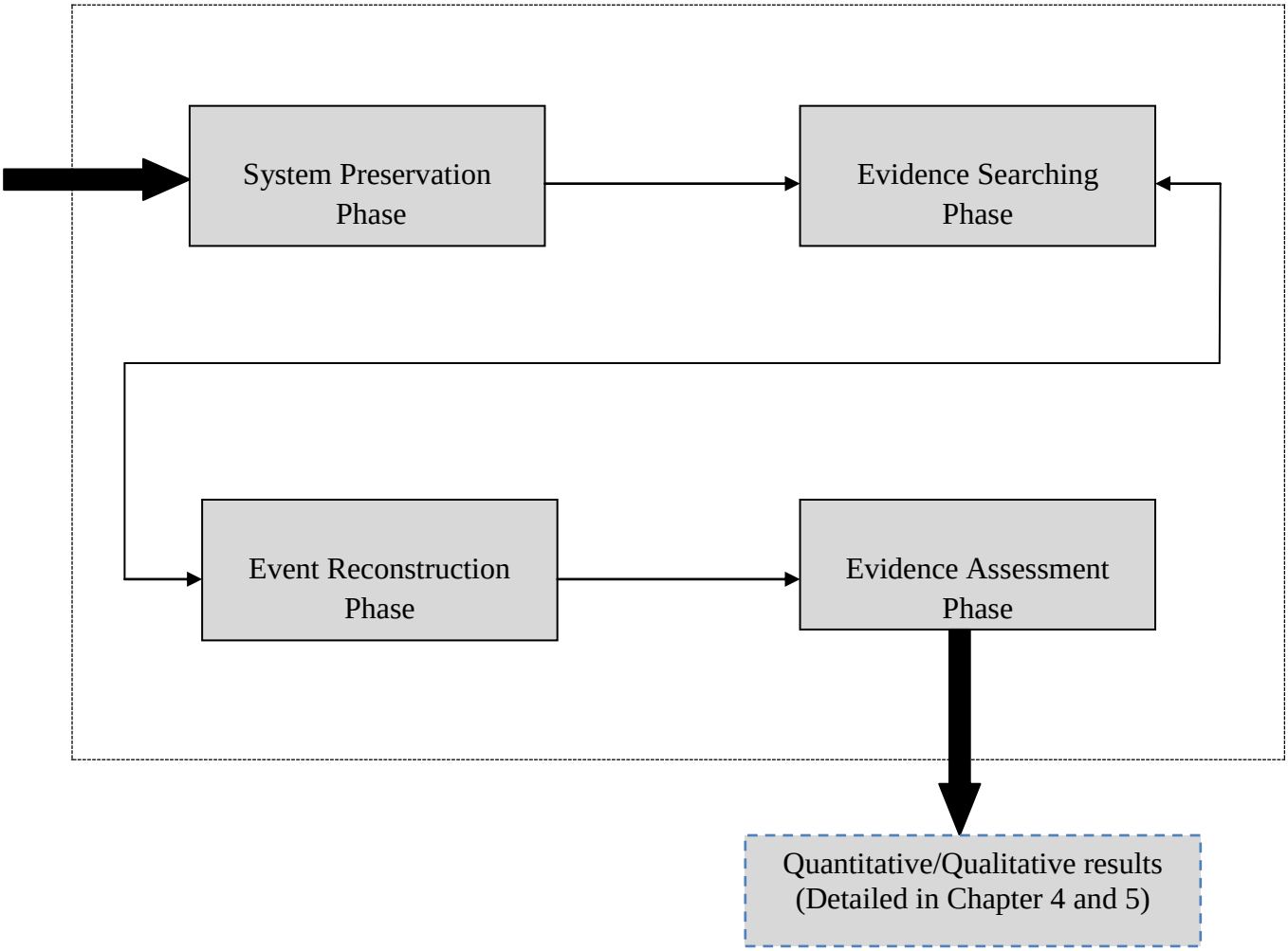


Figure 6.1 Four phases of digital investigation process of application level information

6.3. Steps implemented to design the four phases of digital investigation framework

In this research, the steps implemented to design the four phases of digital investigation process of application level information are as follows:

6.3.1 Step 1 - Identify existing frameworks

In this step, the phases for each framework was analyzed, described and presented as detailed in Chapter 2, Section 2.4.1.1.

6.3.2. Step 2 – Four phases of investigation process

In this section the summary of the four phases of digital investigation process is described based on the appropriate activities/processes and output of the research work of application level information.

6.3.2.1. System Preservation

- Data/image collection of volatile memory on Windows applications.
- Determine what the particular process of the applications is, and identify the possible sources of the data.
- Ensure the integrity and authenticity of the digital evidence.
- Package, transport and store the digital evidence.
- Determine the memory dumping of the applications; extract and convert to a human understandable narrative.
- Duplicate or make copies of the digital evidence captured using standardised and accepted procedures.
- Ensure the validity and integrity of digital evidence for later use.
- Determine the scenarios for the digital investigation process of the research project.
- Determine what user input can be recovered and the time memory of the user input stored on the applications.

6.3.2.2. Evidence Searching

- Determine how the data is produced based on what user is typing; what the user has been doing and what the user has been using the application for.
- Extract and convert the user input from the memory dumped into strings and test the validity of the evidence found.
- Perform the extraction process using a program code, as it was written to extract only the hexadecimal bytes of information within the range 32_{10} to 126_{10} inclusive (i.e. only numbers, Latin characters and punctuation, not any control characters or characters from other languages).
- Determine pattern for searching evidence on the applications which includes pattern of when the original user input is known and when the user input is unknown using some commonly used English words.
- Determine the quantitative metrics and qualitative assessment for the searching of user input on application memory.

6.3.2.3. Event Reconstruction

- Recognize obvious pieces of digital evidence based on four scenarios of the research.
- Identify and locate potential evidence, possibly, within the application memory.
- Construct detailed documentation for the reconstruction process of the user input found.
- Determine the fragments of user input found on the application.
- Determine the partial fragments while using the pattern of known information and whole fragment when the user input is unknown while using some commonly used English words.
- Determine the significance of evidence found based on user input stored on the application memory.
- Reconstruct the user input found based on the two pattern searching techniques of known and unknown information.
- Determine the reconstruction of the user input and compare the extracted data of user input based on the two pattern searching techniques that was used.

- Eliminate the duplication of user input found, perform data analysis and build a timeline of user input sensitive to the investigation questions.
- Build a sample fragment in partial and whole fragment of the information.
- Document the findings and steps taken and determine the analysis of the user input found.

6.3.2.4. Evidence Assessment

- Test the theory of the application level information based on the digital evidence.
- Organise the analysis results from the user input found on application memory.
- Present the user input resulting from the searching/reconstruction phase.
- Determine the issues of relevance of the user input found and its reliability.
- Interpret the statistical information/analysis of data from the reconstruction phase.
- Clarify the evidence and document the findings.
- Summarise and provide explanation of conclusions.
- Present evidence in quantitative and qualitative assessment of user input found.
- Attempt to conform each piece of digital evidence and each event based on four scenarios of the research work.
- Disseminate information from the investigation to communicate the relevant findings.
- Determine how and what digital evidence can be presented for evidential purposes in the court of law.
- Evaluate and review the investigation process to identify areas of improvement.
- Close out the investigation and preserve knowledge gained for further work.

6.3.3. Step 3 – Summarisation of output phase name

In this step, the phase name is summarised based on the activities/processes and output analyzed from step 2. Four phases has been named (i.e. Phase 1 – Phase 4) as in Table 6.1.

Table 6.1 Summary phase of digital investigation process of application level information

Phase	Phase name	Output
Phase 1	System Preservation	Volatile Media, Windows systems, Applications, User Input Recorded.
Phase 2	Evidence Searching	Process Identification, User Input Found, Pattern Searching, Commonly Used English Words, Known User Input.
Phase 3	Event Reconstruction	Events Log, Data, Information, User Input Found, Evidence Information
Phase 4	Evidence Assessment	Evidence Result, Evidence Explanation, New Policies and Investigation Procedures, Evidence Found, Investigation Closed.

6.3.4. Summarisation of digital investigation process

In this section, a summary of the digital investigation process is described based on the appropriate activities, processes and output that was carried out on each of the proposed new phase name of digital investigation process of application level information. The summary is as stated below:

- Windows XP systems were provided for the purpose of this experiment
- Concurrently opened seven most commonly used applications on the windows system
- As described in Chapter 3, Section 3.5, various user input and actions were then made on each of the application within a set period of time
- On the target machine, we ran the trusted command shell of Nigilant32 to capture volatile images at every 30 minutes
- The images that were captured are stored on the external disk and copies were made
- Run automated program written in python with volatility tools to dump memory
- Run automated program written in python to extract strings processes for evidence searching

- Run automated program written in python for to extract only hexadecimal bytes of information within the range 32_{10} to 126_{10} inclusive (i.e. only numbers, Latin characters and punctuation, not any control characters or characters from other languages).
- Run automated program written in python for evidence matching processes.
- Run automated executable code to determine series of data base results of findings
- Reconstruct user input based on the two pattern searching techniques adopted as detailed in Chapter 3, Section 3.4.1.
- Present user input result in quantitative and qualitative assessment as detailed in Chapter 4 and 5 of the thesis.

6.4. Summary

The proposed new phase of the digital investigation process of application level information has been presented. Typical illustration of the framework offers a simplified guideline on steps that should be followed in the forensic process of application level information. A clear output for each of the phases was described in the steps implemented for the designs of the four phases of digital investigation process of application level information. These steps enable us to define the proposed framework that can be used in a forensic investigation of application level information. This proposed framework may be useful on various incident cases, digital devices and digital evidence investigation process.

Chapter 7 Results Discussion

7.1. Introduction

This chapter elaborates on the results and analysis of the data presented in Chapters 4 and 5, it particularly identifies how these results are useful for a forensic investigator. The discussion will focus on volatile memory analysis in the context of the four scenarios developed for this research project. The applicability of the theory of application level information in digital forensic will be discussed. The benefit and the key important areas of the research study of application level information will also be presented.

7.2. Scenarios of the research project

The aim of this research project was to extract application level information from the volatile memory of Windows system and to formulise how the extracted application level information can be reconstructed. Following the investigation of user input on the seven most commonly used applications, four scenarios were designed and the results of the experiments were analysed with respect to the quantitative and qualitative assessments of the user input. In the first scenario, user input was made at set intervals when the applications are still opened on the Windows system. The results of the quantitative assessment revealed the percentage of user input found on each of the applications. This information was illustrated in graphs as presented in Chapter 4 and the extracted application level information was reconstructed as described in Chapter 5. There was a large quantity of user input recovered from some of these applications whereas, small quantity of related user input were stored in the memory of other applications. For example, the highest percentage of user input was recovered from the memory of Internet Explorer 7.0. It was discovered that the user input found is stored in contiguous blocks of the application memory, making it possible to recover data in partial and whole fragments.

Using the criteria defined in this scenario, other applications like MS Word, MS PowerPoint and MS Outlook Email also reported a large amount of user input stored in the application memory. Small amounts of user input were found on applications like MS Excel, MS Access and Adobe Reader 8.0 applications. The qualitative assessment of user input was presented with a sample of the extracted application level information. This information has been arranged and rearranged in

a sentence or sentences of information assumed to be the original user input. The user input activities were reconstructed to ascertain what the user was typing on the application, what the user had been doing and what the user might have been using the application for. Using this assessment in digital investigations, forensic investigators should be able to understand what crime related data could be hidden on Windows applications.

In the second scenario of the research project, the user input was assessed based on the time aspect of user input as it was stored in the application memory. As required, user input was made once, while the applications are still running, volatile memory was captured at set intervals. As analysed, a large amount of user input was stored for a long period of time in the application memory. Using this assessment, an investigator should be able to understand how much data is lost on the application memory. The qualitative assessment of user input resulted in the presentation of sample application level information that was extracted and that had been reconstructed to ascertain what the user was typing on the application and how this information can be arranged and rearranged to form a sentence or sentences of the original user input. Following the series of experiments that have been carried out, an investigator will be able to understand that the user input can be stored for a long period of time in the memory. For example, MS Word, MS PowerPoint, MS Outlook Email, MS Internet Explorer 7.0, MS Excel and Adobe Reader 8.0. It is also known that MS Access contains only a small amount of user input in the application memory.

In the third scenario of the research project, the time in memory of user input was investigated when the applications were closed and the Windows system is not used for any other purpose. In this scenario, user input was made once and volatile memory was captured at set intervals. The result of the quantitative assessment was different to the previous scenario because a large amount of user input was lost. Only two (i.e. MS Outlook Email and MS Internet Explorer 7.0) out of the seven most commonly used applications retained user input for a long period of time, while less user input was found in the memory allocated to other applications like MS Word, MS PowerPoint, MS Excel, Adobe Reader 8.0 and MS Access. The qualitative assessment of user input activities was investigated and a sample of the application level information was extracted from the memory and then reconstructed to ascertain what the user might have been doing on the

applications. A series of experiments were carried out, some related data of user input were arranged and rearranged to form complete or incomplete sentences. By assessing this scenario, a forensic investigator would be able to understand the user input stored in the memory and how much data is lost in the application memory.

In the fourth scenario of the research project, the time in memory of user input was found with a different approach. In this scenario the user input was made once and then the applications were closed, while the system was used to run other different applications. Volatile memory was captured at set intervals. The quantitative assessment of user input indicated that a large amount of user input was lost in the memory over a period of time when the applications were closed. Again, two applications (MS Outlook Email and MS Internet Explorer) out of seven applications contain a significant amount of user input in the memory. User input was stored for a longer period of time. Less data was stored in the memory of other applications like MS Word, MS PowerPoint, MS Excel, Adobe Reader 8.0 and MS Access. But, as compared to the previous scenario 3, it was discovered that user input was dispersed more in the memory. This may be because a large amount of memory may have been lost when the Windows system is used to run other applications.

The original user input may have been overwritten by other information when the system was used to run other applications. The qualitative assessment of user input was presented with sample application level information that was extracted from the application memory after it had been reconstructed. This information was arranged and rearranged to form a sentence or sentences of user input.

The results and the analysis taken together from Chapter 4 and Chapter 5 should allow a forensic investigator to determine what would provide the most valuable information about the user input that can be recovered from the memory of these applications. The analysis also assists in determining the relevant user input activities that have been reconstructed on each of the applications based on four scenarios of the research project.

7.3. Metrics of the experiment in quantitative assessment

In this research project, there are four metrics that were designed for the quantitative assessment of user input on applications. The four metrics include the length of the user input, the number of times a character of the user input is repeated, the percentage of the user input found and the length of user input that has been found in continuous blocks of the application memory.

The first metric is the length of user input stored in the memory of each of the applications. This information may be useful to forensic investigators to determine the original user input made on Windows system.

The second metric is the mean number of times each character of user input is repeated in the application memory. This information may be useful to forensic investigators to determine the numbers of time the user input characters are repeated which, of course, will help to determine what the user was typing on the application.

The third metric is the percentage of user input found in the application memory. Forensic investigators should be able to use this metric to determine what user input is made on the application and how much of that information was still residing in the memory while extracting the user input.

The fourth metric is the mean length of user input found in continuous blocks of the application memory. This metric may be useful to a forensic investigator to find out on how easy it is to find the original user input (longer blocks of user input provide more contextual information).

Forensic investigators should be able to use the quantity of user input found to determine the amount of the original user input stored in the memory of each of the applications. This information may also be useful in the recreation of user input activities on the application memory.

7.4. Qualitative assessment of reconstructing user input

The qualitative assessment of user input was presented in Chapter 5. The reconstruction of user input activities on four scenarios of the research project indicates that the reconstruction of user input on scenario 1 and 2 was easily achieved but it was difficult in scenario 3 and 4 except on applications like MS Outlook Email and MS Internet Explorer where easy reconstruction of user input activities was noted. It was discovered that user input is dispersed in the application memory, but some related data of user input was found stored in continuous blocks. By reconstructing user input, information is arranged and rearranged to find sentences of the original user input found in the application memory.

It may be interesting to a forensic investigator that user input can be recovered and this information may be presented as partial or whole fragments. The user input can be stored in continuous blocks of the application memory. Throughout the experiments carried out, it can be said that the user input on some applications was reconstructed easily while, in other applications, it was proved to be very difficult. Forensic investigators should be able to understand that the reconstruction of user input activities can only be made possible when related data of user input were recovered from the memory.

The user input stored in continuous blocks of the application memory can be reconstructed easily because the information can be arranged in whole fragments. For example, the user input on applications like MS Word, MS Outlook Email, MS Internet Explorer and MS PowerPoint were stored in whole fragments. However, the user input may be arranged in partial fragment when less data of user input were recovered. For example, user input on applications like MS Excel, MS Access, and Adobe Reader 8.0 contains more partial fragments making the reconstruction of user input activities difficult.

7.5. Good forensic practice for application level information

It may be recommended to forensic investigators that when assessing the quality of the extracted application level information of user input activities on Windows applications, all steps are taken to document it properly. It is expected that an investigator should maintain proper documentation and the processes to maintain proper documentation must be enough to carry out investigations,

copies of documents related to the extraction and the reconstruction of application level information must be made. A backup maintenance of this information must be prepared and produced on external devices. A good storage location of the devices must be maintained and this information must be preserved for future use.

7.6. Ability of the tools used for application level information

There are numbers of tools that can be used to acquire volatile memory images of Windows systems. For example, a research paper of (Iain, Jon, Theodore, & Andrew, 2008) presents on acquiring volatile operating system data tools and techniques. The paper emphasizes the importance of understanding the potential value of volatile data and how best to collate forensic artefacts to the benefit of the investigation. To ensure the preservation and integrity of the information in this research project, the tool used in capturing the volatile memory of Windows applications was Nigilant32 (Matthew, 2008). This tool allows an investigator to preview a memory image and take a snapshot of it. The time and file system impact for memory acquisition revealed that Nigilant32 has a small footprint, using less than 1 MB in memory when loaded and with a minimal impact during acquisition.

7.7. The importance of volatile memory analysis of application level information

A seminar of (NIST, 2008) addressed the need for more sophisticated tools to aid volatile memory acquisition and analysis. This will help in the assessment of external and internal intrusions, which may continue even in the robust security infrastructures of the best government and industry systems. The key to successfully preventing and responding to any digital fraud investigations is the sound identification, collection, preservation and analysis of computer evidence, for example from the volatile memory of Windows computer systems. The volatile data may be vital to determine criminal activity on a computer. In this research project, the manner in which user input is stored on Windows applications was investigated to identify the usefulness of the application level information and volatile memory analysis. This research into volatile memory analysis of application level information indicates its major importance in digital forensics and its possible usefulness as evidence in the court of law. Following the investigations of the four scenarios of the research project, a forensic investigator may begin to understand that the user input was recovered from the memory allocated to the applications.

According to a research of Volatools (Aaron & Nick, 2007), which focused on the integration of volatile memory forensics, current data analysis is limited to the extraction of cryptographic key material from volatile memory. Therefore, the research of application level information from volatile memory is a critical component of a digital crime scene. In this research project, the four metrics designed can be used in the quantitative assessment of user input stored on the applications. This information may be useful to forensic investigator as it takes an in-depth view of what the user input can be recovered from the application memory. The quantity of user input recovered was recorded using the technique of searching for the user input stored in the memory of the applications. It was found that when the pattern of the original user input of known information was used, there is a large amount of user input found in the memory but in some applications there are less data of user input that was found. When the pattern of some commonly used English words was used, there are large amount of user input that was found and also in some other applications, less data of user input was found. This information is detailed in Chapter 4.

The pattern searching techniques as detailed in Chapter 3 may be useful to forensic investigators because the pattern of the original user input gives some useful and details information of user input that was stored in the memory. Also, it can be said that most of the user input was found using this technique of known information about the original user input. The pattern of some commonly used English words is unique on its own capacity and the information found may be useful to forensic investigators because it gives concise information about the user input stored in the memory of each applications. In this research project, the two pattern matching techniques of searching for user input gives a complete picture of information stored on the application memory but, it can be said that the results of the user input that was recovered using these techniques varied in some other applications. The results of chapters 4 and 5 show that application level information from each of the applications may be useful in digital investigations. The user input recovered describes how the extracted application level information was reconstructed to form sentences of the original user input. Forensic investigators should be able to understand that the approach of investigating application level information may be applied while investigating user input activities on other social networking and internet applications like Facebook and Twitter.

7.8. Procedure formulized for application level information

In this research project, a procedure has been formulised for the extraction and reconstruction of user input from the volatile memory of applications. This procedure fulfils the research aim and objectives. The forensic investigation processes that were formulized are as follows:

- Capture the volatile memory using Nigilant32
- Use Volatility software to dump the memory that has been allocated to the applications of interest
- Use a simple python program to extract the characters of user input in the expected format
- Use a simple program to search for user input
- Reconstruct the user input

7.9. Applicability of the theory of application level information

The area of application level information and volatile memory analysis of Windows applications have been investigated and the applicability of this research as evidence in a court of law may be possible. This information may be useful to forensic investigators because the investigation procedures and guidelines to be used by forensic investigators have been provided in section 6.8 of this chapter. Based on the experiments carried out on applications, some related data of user input may be stored in the memory of some commonly used Windows applications. The five Daubert tests (Daubert v. Merrell Dow Pharmaceuticals, 1993) were applied to justify the major contributions of this research project, to assess the admissibility of application level information as a scientific method of gathering data and to be used as evidence in the court of law. Three out of the five Daubert tests of scientific standards of gathering evidence from digital devices were achieved.

7.9.1. Benefit of application level information to forensic investigator

The forensic investigator may be able to use the application level information gathered from volatile memory of Windows computer systems as evidence information in the court of law but, the application level information must be collected in a way that is legally admissible. By using the two pattern matching techniques when searching for user input in the volatile memory,

known and unknown information of user input can be found. Although the pattern searching techniques for finding data have not been generally accepted by the scientific community, the approach of searching for data was a unique element of this research project and it is anticipated to become a useful tool in future forensic investigations. For example in real life, a forensic investigator may apply the research project scenario by capturing a sample application running or closed on Windows system.

A conversion program was written which would extract only those hexadecimal bytes of information within the range 32_{10} to 126_{10} inclusive (i.e. only numbers, Latin characters and punctuation, not any control characters or characters from other languages). Following this, is the analysis of the extracted application level information to give the answer to what the user is typing, what the user has been doing and what the user has been using the application for.

The pattern matching techniques can be used to search for user input stored on the applications when the original user input was known and when user input is unknown, some commonly used English words were used to search for user input stored on volatile memory of the applications. The analysis result of quantitative and qualitative assessment of user input may be presented. This information may be useful to forensic investigators while investigating volatile memory data.

7.9.2. Key Important area of the research study of application level information

The research theory of application level information has contributed to the scientific community of digital forensic in some unique ways as follows.

- First, the scientific theory of application level information has been empirically tested.
- Second, the scientific theory of application level information and the technique used in the search for user input have been subjected to peer review and publication as listed in Appendix A.
- Third, the quantitative amount of user input found on commonly used applications were presented as detailed in Chapter 4.

- Fourth, the technique used in searching for user input on applications has not been maintained by any existing standards.
- Fifth, the theory of application level information has not yet been generally accepted in the community but, the theory may be useful to forensic investigators. It can be said that the technique and the results can be presented with sufficient clarity and simplicity to the court of law for a jury or judges' understanding. The qualitative assessment of user inputs can only be made successful by reconstructing the user input recovered from the application memory. Furthermore, the research theory of application level information has made some unique contributions to the scientific methods of gathering evidence from digital devices.

7.10. Summary

This chapter has discussed the application level information. The research investigation was based on four scenarios of user input on application memory. The procedures formulized for the investigation can be used by forensic investigators to ascertain what the user typing, what the user has been doing and what the user is using the application for. It is suggested that in the area of digital forensics, application level information may be useful in crime investigation.

Chapter 8 Conclusions and Further Work

8.1. Introduction

This chapter summarises the research carried out and the major findings of this thesis. Future work, based on this research, and the author's contributions are outlined.

8.2. Conclusions

The scenarios developed in this thesis work have attempted to show the usefulness of application level information that was recovered from the volatile memory allocated to the Windows applications. The analysis of user input indicates that related user actions can be found in volatile memory of live computer systems either when the applications are still running or closed. It was discovered that once a user inputs information on the system, whether the application is closed or kept open, the related user input can lie there for some time. Based on the experiments carried out for this research project, it can be said that if volatile memory is not captured during crime scene preservation, then it might not be possible to retrieve valuable information that could be useful to forensic investigators.

Volatile memory analysis may be time consuming to recover user input, but the related user input recovered may result in useful information to a forensic investigator. It has been found that the application level information can be retrieved as both partial and whole fragments. This information may picture the digital event of user input activities using the applications on the computer system. The information recovered from the four scenarios may be useful to forensic investigators because the four metrics used in the quantitative assessment of user input detail how much data can be recovered and it also shows how user input may be found in the memory allocated to an application.

In addition, the information recovered from the qualitative assessment of user input based on the four scenarios may also be useful to forensic investigators because the results of the user input extracted and reconstructed can be used to infer what the user was typing on the applications, what they have been doing and what they are using the applications for.

The approach used in the investigation of the user input stored on the application memory may be useful to forensic investigators. The pattern technique of using some commonly used English words could, of course, be replaced by specific words that are related to an investigation. This may become part of standard forensic analysis in digital investigation.

8.3. Discussions and contribution

The objectives of the research were to investigate the user input stored in the volatile memory of application in order to uncover information that may have previously been "hidden" to forensic investigators. This was achieved by extracting application level information from the volatile memory of Windows systems and formulising how the extracted application level information can be reconstructed to describe what user activities had taken place on the applications under investigation. This research has focused on two techniques for recovering user input; pattern matching when the original user input is known and pattern matching with commonly used English words. To undertake the study, commonly used applications were identified by contacting businesses and the procedures for capturing volatile memory were designed, built and tested for the four different scenarios of the research project.

A computer system running Windows XP with service pack (SP2) was used to capture 100 images of volatile memory at set intervals of 30minutes for each application based on four scenarios of the research. The pattern matching algorithm allows investigations on what the user is typing on the applications, what they have been doing and what they have been using the application for.

The results of the experiments based on four scenarios were discussed and presented in quantitative and qualitative assessment of user input in Chapter 4 and 5. From the discussion of the results in Chapter 6 and the main contributions to knowledge with the key achievement of the research work that is presented in Section 7.5 below, it is suggested that application level information and volatile memory analysis may become an important source of information in a digital investigation.

8.4. Evaluation of the thesis work

In this thesis, the research project was conducted so as to evaluate the approaches proposed. Many experiments was carried out on seven most commonly used Windows applications based on this formulised approaches. Yet, these approaches call for further experimentation on other Windows operating systems.

- In scenario 1, user input was easily recovered from the applications like MS Word, MS PowerPoint, MS Outlook Email and MS Internet Explorer 7.0 while little information was recovered from the applications MS Excel, MS Access and Adobe Reader 8.0. How can other method be used to provide more related data of user input where little information was recovered on the applications?

- In scenario 2, a significant amount of user input was found from all the applications except the MS Access. How can the extension of this research work be applied to other Windows systems in order to get the same results?

- The result is not the same when the applications have been closed and the user is not interacting with the system in scenario 3. It was discovered that only on Outlook Email and Internet Explorer 7.0 gave a significant amount of user input. The rest of the applications resulted in less data of user input that was found in the memory. How can the extension of this research work be applied to other applications when a Windows system is in hibernate mode? What type of information can be collected?

- In scenario 4, the result was changed when the application was closed and the system is used to run other applications, a reduced amount of user input was found in the memory. This information can be retained in the memory for a long period of time, but was only found in the memory allocated to Outlook Email and Internet Explorer 7.0. In other applications, user input was retained for a shorter period of time and the amount found was reduced as compared to the previous scenarios. How can the extension of this research work be applied to other social networking application when a Windows system is in safe mode? What type of information that can be recovered? Or can we get the same results as it was shown in this scenario4?

8.5. Thesis contributions

The research contribution to knowledge is based on the extracted application level information from the volatile memory of Windows applications and this was reconstructed to infer the following: "what the user was doing on the applications?", "what the user has been doing?" and "what the user was using the application for?".

The main contributions given below include a list of the key achievement of this research work:

- The contribution to academia/society includes:
  1. For the first time, a definition of application level information from investigating the volatile memory and analysis of user input on applications was presented.
  2. The quantitative and qualitative assessment of application level information on commonly used applications that forms the digital forensic analysis were presented.
  3. A conversion program was written to extract only hexadecimal bytes of information within the range 32_{10} to 126_{10} inclusive (i.e. only numbers, Latin characters and punctuation, not any control characters or characters from other languages). For further detail see page 41.
  4. The novel techniques of the two pattern matching techniques of when user input is known and when user input is unknown by using some commonly used English words are used for searching and reconstructing user input activities on volatile memory of applications.
  5. A study of application level information based on four scenarios of the research project was designed. For further detail see Chapter 3.
  6. An approach to the extraction of forensically relevant application level information from the volatile memory of Windows application was designed. See Section 6.8.

- The contribution to the industry includes:
  7. The research theory of application level information has made some unique contributions to the scientific methods of gathering evidence from digital devices.

8. The approach of the four event-based scenarios of the research work as discussed in Chapter 3 may be applied or useful in digital crime investigation
9. The procedure formulised for application level information in Section 6.8 may be applied to other internet applications like Facebook, twitter or chatting blogs where information contained in the memory allocated to these applications may be used for solving crime and tracing fraud.

8.6. Future work

The research project uncovered information that may be hidden to forensic investigators on these commonly used applications and the usefulness of the extracted application level information from volatile memory of Windows application was made possible based on four scenarios of the research methods. The research work also raises other issues which could be explored in future research.

- Further work should involve the investigation of user input on other operating systems like Windows Vista and Windows 7 which are quite new to the market. This would serve the forensic community in the investigation of user input that can be recovered from the memory of these two operating systems (Windows Vista and Windows 7). The same four scenarios may be used and applied to find different behaviour in those two systems. The research has covered the states of the application on a live Windows system, but it could be extended to when the system is in safe mode or hibernate states. This research has found that if the application has been closed then a smaller amount of data can be recovered from the allocated memory. Further research is required to identify if there is any possibility that user input can be recovered from closed applications if the system is in the hibernate or safe mode state when volatile memory is captured.

- Another extension of this work would be to extend it to 64 bit operating systems of Windows Vista and Windows 7. In addition to this, very little has been done on combining both the application memory and page file information in digital forensics. This could be of great value to the forensic community because incorporating memory

and page file may give a complete picture of the whole crime scene. This could be a very interesting research project and become a valuable work.

- At the moment, there is no standard way of acquiring volatile memory and page file information and the impact of the available tools is varied. A standardized solution to this problem could be a valuable contribution. Finally, assessing the impact of the currently available tools and techniques will serve the forensic community well.

References

AAron, W. (2008, May 5). *The Volatility Framework: Volatile memory artifact extraction utility framework.* Retrieved October 24, 2009, from Volatile Systems, LLC: https://www.volatilesystems.com/default/volatility

AAron, W., & Nick, L. P. (2007, February). Volatools: Integrating volatile memory into the digital investigation process. *Journal of Digital Investigation, II*(2), 26-35.

AAron, W., T. Fraser, N. L., & William, A. A. (2006, December). FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory. *Digital Investigation Journal, 3*(4), 197-210.

AccessData. (2008, April 2). *AccessData*. Retrieved June 11, 2009, from AccessData digital forensics software at a glance. Forensic Toolkit® (FTK®): http://accessdata.com/products/computer-forensics/ftk

ACPO. (1998, April 02). Retrieved August 15, 2010, from Good Practice Guide for Computer-Based Electronic Evidence: http://www.7safe.com/electronic_evidence/ACPO_guidelines_computer_evidence.pdf

Andreas, S. (2006). Searching for processes and threads in microsoft windows memory dumps. *Digital Investigation, 6*(10), 10-16.

Andreas, S. (2008). The impact of Microsoft Windows pool allocation strategies on memory forensics. *Digital Investigation, 5*(7), 58-64.

Andrew, C., Andrew, C., Lodovico, M., Golden, G. R., & Vassil, R. (2008). FACE: Automated digital evidence discovery and correlation. *Digital Investigation, 5*(8), 65-75.

ASCII. (2007, March 26). *Hexadecimal bytes*. Retrieved October 30, 2010, from ASCII Table and Description: http://www.asciitable.com/

Autonomy. (2008, May 8). *Autonomy's Meaning Based Computing Infrastructure.* Retrieved August 11, 2011, from Autonomy's Advanced Search and Scalability Solutions for SharePoint: http://publications.autonomy.com/pdfs/Power/White%20Papers/SharePoint/Autonomy_Search_Scalability_for_SharePoint_WP.pdf

Baryamureeba, V., & Tushabe, F. (2004). The Enhanced Digital Investigation Process Model. *Proceeding of Digital Forensic Research Workshop. 5*, pp. 1-9. Baltimore, MD.: Citeseer.

Betz, C. (2005, August 17). *DFRWS 2005 Forensics Challenge Memparser Analysis Tool.* Retrieved January 16, 2010, from DFRWS 2005 Forensics Challenge: http://www.dfrws.org/2005/challenge/memparser.shtml

Brendan, D.-G. (2008). Forensic analysis of the Windows registry in memory5. *Digital Investigation, 5*(3), 26-32.

Brendan, D.-G., Abhinav, S., Patrick, T., & Jonathon, G. (2009, March). Robust signatures for kernel data structures. *Proceedings of the 16th ACM conference on Computer, 9*, pp. 1-12. Chicago, Illinois, USA.

Brian, C. (2005, August 10). *Digital Forensics Tool Testing Images.* Retrieved July 14, 2010, from Digital Forensics Tool Testing Images: http://dftt.sourceforge.net/

Brian, C. (2005). *File System Forensic Analysis* (1st ed.). Upper Saddle River, NJ, U.S.: Addison-Wesley, Pearson Education, Inc. .

Brian, C. D., & Eugene, S. H. (2004). Defining event reconstruction of digital crime scenes. *Forensic Sciences, 49*(6), 1291-1298.

Brian, C. D., & Joe, G. (2004). A hardware-based memory acquisition procedure for digital investigations. *International journal of digital evidence (IJDE), 1*(1), 10-23.

Brian, C., & Eugene, H. S. (2003, February). Getting Physical with the Digital Investigation Process. *International Journal of Digital Evidence, 2*(2), 4-8. Retrieved from Incident Response &.

Brian, D. C., & Eugene, H. S. (2004, February). Event-based Digital Forensic Investigation Framework. *Digital Investigations, 1*(1), 1-12.

Brian, D. C., & Eugene, H. S. (2005). Automated digital evidence target definition using outlier analysis and existing evidence. *2005 Digital forensic research workshop (DFRWS)* (pp. 1-10). New Orleans, LA: Elsevier, Inc., The Journal of Digital Investigation.

Brian, D. C., & Eugene, H. S. (2006). Categories of digital investigation analysis techniques based on the computer history model. *Digital Investigation, 6*(11), 121-130.

Brian, H., & Kara, N. (2008, April). Forensics examination of volatile system data using virtual introspection. *ACM SIGOPS Operating Systems, 42*(3), 1-9.

Butler, J. M. (2007, January). Forensic Application Validation: What is it, Why Does it Matter and How Should it be Done. *International Journal of Forensic Sceince, 52*(2), 12-45.

Cameron, H. M., Eoghan, C., & James, M. A. (2007). *Malware Forensics: Investigating and Analyzing Malicious Code* (1 ed.). (J. Eby, Ed.) Burlington, USA, MA: Syngress Publishing.

Carrier, B. D. (2006). *A Hypothesis-based Approach to Digital Forensic Investigations.* West Lafayette: CERIAS Tech Report 2006-06, Purdue University, Center for Education and Research in Information Assurance and Security.

Chris, P., & Kevin, M. (2003). *Incident response and computer forensics* (2nd ed.). Emeryville, Califonia 94608: McGraw-Hill/Osborne.

Columbo. (2010, June 11). *ColumboForensicstm. Products and Applications*. Retrieved March 2, 2011, from ColumboForen.ColumboForensicstm: http://www.gillcgroup.com/products/forensics/

Corporation., M. (2008, March 7). *Microsoft Corporation.Bruce Sanderson's general Windows information: RAM, virtual memory, pagefile, and all that stuff.* Retrieved December 5, 2009, from Microsoft Corporation. RAM, virtual memory, pagefile, and memory management in Windows: http://support.microsoft.com/kb/2160852

Dan, F., & Wietse, V. (2005). *Forensic Discovery* (1st ed.). US: Addison-Wesley Professional.

Daubert, v., & Merrell Dow Pharmaceuticals, I. (1993, June 28). *Prerequisites For The Admissibility Of Expert Witness Testimony Under Federal Law: An Examination Of Daubert v. Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579 (1993) And Its Progeny.* Retrieved December 22, 2010, from Daubert v. Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579, 125 L. Ed.2d 469, 113 S. Ct. 2786: http://www.eastmansmith.com/documents/publications/Daubert%20seminar-materials.pdf, http://www.hcs.harvard.edu/~jus/0302/orofino.pdf

David, B. (2001, August 7). *"Digital Forensics," by David Baker.* Retrieved July 20, 2010, from A Road Map for Digital Forensic Research DFRWS TECHNICAL REPORT DTR - T001-01 FINAL: http://www.dfrws.org/2001/dfrws-rm-final.pdf

DFRWS. (2001). *DFRWS TECHNICAL REPORT A road map for digital forensic research.* Utica, New York: DTR - T001-01 FINAL. Report From the First Digital Forensic Research Workshop (DFRWS).

DFRWS. (2007, July 3). *DFRWS 2007 Annual Conference.* Retrieved November 11, 2009, from Digital Forensic Research Workshops (DFRWS): http://www.dfrws.org/2007/challenge/index.shtml

Eoghan, C. (2007, March). Digital evidence maps - A sign of the times. *Digital Investigation, 4*(1), 1-2.

Eoghan, C. (2007, June). What does "forensically sound" really mean. *Digital Investigation, 4*(2), 49-50.

Eoghan, C., James, M. A., & Cameron, H. M. (2008). *Malware Forensics: Investigating and Analyzing Malicious Code* (1st ed.). (J. Eby, Ed.) Burlington, MA, USA: Syngress Publishing, Inc.

Eric, V. B., & Vincent, T. L. (2006, February). Digital Evidence: Challenging the presumption of reliability. *Journal of Digital Forensic Practice, 1*(1), 19-26.

Erin, E. K., M.F.S, & J.D. (2005, May 1). *Confluence of digital evidence and the law: On the forensic soundness of live-remote digital evidence collection.* Retrieved April 28, 2011, from UCLA Journal of Law & Technology: http://www.lawtechjournal.com/articles/2005/05_051201_Kenneally.pdf

Felix, C. F., & Bastian, S. (2007). A common process model for incident response and computer forensics. *Proceedings of Conference on IT Incident Management and IT Forensics. 114GI*, pp. 19-40. Germany: IMF.

Frederick, B. C. (2006). Challenges to digital forensic evidence. *Cybercrime Summit 06. Retrieved from: http://all.net/Talks* (pp. 21-45). Washington: Digital Forensic Research Workshop.

Frederick, B. C. (2008, May). Fundamentals of digital forensic evidence. *Journal of Digital Investigation, II*(2), 1-27.

Gabriela, L. G. (2007). Forensic physical memory analysis: an overview of tools and techniques. *TKK T-110.5290 Seminar on Network Security* (pp. 1-6). Helsinki, Finland: TKK T-110.5290 Seminar on Network Security 2007-10-11/12.

Gary, C. K. (2010, September 1). Judges' awareness, understanding, and application of digital evidence. Nova Southeastern University, Fort Lauderdale-Davie , Florida, USA.

George, M. G. (2007, November 30). *FAU:Forensic Acquisition Utilities*. Retrieved March 17, 2010, from Forensic Acquisition Utilities: http://gmgsystemsinc.com/fau/

George, M. G., & Robert-Jan, M. (2005, August 17). *DFRWS 2005 Forensics Challenge Kntlist Analysis Tool.* Retrieved April 8, 2010, from DFRWS 2005 Forensics Challenge: http://www.dfrws.org/2005/challenge/kntlist.shtml

Guidance Software. (2008, May 3). *EnCase® Forensic.* Retrieved May 3, 2010, from Guidance Software. EnCase® Forensic: http://www.guidancesoftware.com/computer-forensics-ediscovery-software-digital-evidence.htm

Harlan, C. (2009). Windows Forensic Analysis DVD Toolkit DVD Toolkit 2E. In C. Harlan, *Windows Memory Analysis* (2nd ed., pp. 89-155). Burlington, MA: Syngress Publishing, Inc.

Harlan, C., & Dave, K. (2007). *Windows forensic analysis including DVD toolkit. Incident response and cybercrime investigation secrets* (1st ed.). Burlington, MA, Windows Forensic Analysis Including DVD Toolkit: Syngress Publishing, Inc.

Harlan, C., Dave, K., & Jesse, K. (2007). *Windows Forensic Analysis: Incident Response and Cybercrime Investigation Secrets*, (2 ed.). (J. Eby, Ed.) Burlingtton, USA, MA: Syngress Publishing, Inc.

Harlan, C., Eoghan, C., & Cameron, H. M. (2007). *Malware Forensics - Investigation and analyzing malicious code* (3rd Edition ed.). (J. E. Curtis W.Rose, Ed.) Burlington, MA, USA: Syngress Publishing, Inc.

Hejazi, S., Talhi, C., & Debbabi, M. (2009, October). Extraction of Foresnically Ssensitive Information From Windows Physical Memory. *Journal of Digital Investigation, 6*(3), 121-131.

Home Office UK. (2010, February 20). Retrieved July 12, 2011, from Home Office Forensic Science Regulator: http://www.homeoffice.gov.uk/publications/police/forensic-science-regulator1/fs-review-researchers?view=Binary

Home Office UK. (2010, June 9). Retrieved March 14, 2011, from The Forensic Science Service Order By Houese of Common: http://www.publications.parliament.uk/pa/cm201012/cmselect/cmsctech/855/855.pdf

IAAC. (2007, March 2). *Directors and Corporate Advisors.* Retrieved August 14, 2010, from Guide to Digital Investigations and Evidence Information Assurance Advisory Council: http://www.iaac.org.uk/LinkClick.aspx?link=Evidence+of+Cyber-Crime+v12 rev.pdf&mid=680&contenttype=application/pdf, [Viewed 31/07/2011]

IAAC. (2009, January 6). *Directors and Corporate Advisors: Information Assurance Advisory Council (IAAC) Directors and corporate advisors.* Retrieved March 16, 2011, from Directors and Corporate Advisors (IAAC).Guide to Digital Investigations and Evidence: http://www.iaac.org.uk/_media/DigitalInvestigations2009.pdf

Iain, S., Jon, E., Theodore, T., & Andrew, B. (2008, April). Acquiring volatile operating system data tools and techniques. *ACM SIGOPS Operating Systems Review*, *42*(3), 65-73.

James, O. H., Paul, H. L., Jessica, R. S., & Joseph, J. S. (2010, January). Electronic Discovery:Evidence Chain of Custody and Control. In E. C. Cory Altheide, *Handbook of Digital Forensics and Investigation* (pp. 63-134). New York: Elsevier.

James, O., & Gilbert, L. P. (2010). Windows operating systems agnostic memory analysis. *Digital Investigation, 5*(7), 48-56.

James, O., & Gillbert, L. P. (2011, August). Extracting the Windows Clipboard From Physical Memory. *Journal of Digital Investigation, 5*(14), 118-124.

Jason, S., Ewa, H., Derek, B., & Magdalena, S. (2007, June). User data persistence in physical memory. *Digital Investigation, 4*(2), 68-72.

John, M. B. (2007, January 22). *Standards and Guidelines Regarding Validation, Validation: What Is It, Why Does It Matter, and How Should It Be Done?* Retrieved June 25, 2011, from Applied Biosystem, Forensic News, Validation Information to Aid Forensic DNA Laboratories: http://marketing.appliedbiosystems.com/images/forensic/volume8/PDFs_submitted/02A_CustomerCorner_Val_What_is_it.pdf

John, W. W. (1984, April). Scientific Evidence and the Question of Judicial Capacity Folkes v. Chadd, 99 Eng. Rep. 589 (1782). *William and Mary Law Review, 25*(8), 600-659. Retrieved from Folkes v. Chadd, 99 Eng. SCIENTIFIC EVIDENCE AND THE QUESTION OF JUDICIAL CAPACITY: http://scholarship.law.wm.edu/cgi/viewcontent.cgi?article=2214&context=wmlr&sei-redir=1#search=%2299%20Eng.%20Rep.%20589%20%281782%29.%22

Joseph, S., Shari, S. D., & Neil, V. (2002, October). Legal Perceptions of Science and Expert Knowledge. *The American Psychological Association, Inc., 8*(2), 139-153.

Joshua, J., Pavel, G., Mohd, T. A., & Yuandong, Z. (2009, September). Analysis of Evidence Using Formal Event Reconstruction. *ICDF2C* (pp. 12-26). Albany, NY, USA: First International Conference on Digital Fonrensics and Cyberc Crime. Retrieved from J. James, P. Gladyshev, M.T.Abdullah, Y. Zhu "Analysis of Evidence Using Formal Event Reconstruction", in Proceedings of the First International Conference on Digital Fonrensics and Cyberc Crime (ICDF2C), Albany, NY, USA, September 2009.

Karen, K., Suzanne, C., Tim, G., & Hung, D. (2006). *Guide to Integrating Forensic Techniques into Incident Response, National Institute of Standards and Technology.* Gaithersburg, MD : NIST Special Publication 800-86 .

Law, C. (2009, July 1). Retrieved September 25, 2010, from The Admissibility of Expert Evidence in Criminal Proceedings in England and Wales - A New Approach to the Determination of Evidentiary Reliability: http://www.justice.gov.uk/lawcommission/docs/cp190_Expert_Evidence_Consultation.pdf

ManTech. (2008, August 15). *Memory dd*. Retrieved March 21, 2010, from ManTech international corporation: http://www.mantech.com/msma/MDD.asp

Marcus, K. R., James, G., Rick, M., Timothy, W., & Steve, D. (2006). Computer Forensics Field Triage Process Model. *Proceedings of Conference on Digital Forensics, Security and Law* (pp. 27-40). Digital Forensics, Security and Law.

Mariusz, B. (2007, March 14). *Windows Memory Forensic Toolkit: Finding digital evidence in physical memory*. Retrieved August 28, 2011, from Finding digital evidence in physical memory: http://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Burdach/bh-fed-06-burdach-up.pdf

Mark, R. C., Reith, M., Clint, C., & Gregg, G. (2002, January). An examination of digital forensic models. *International Journal of Digital Evidence, I*(3), 1-12.

Matthew, M. S. (2008, March 20). *Agile Risk Management Nigilant32 Windows afterdark forensic*. Retrieved February 2, 2010, from Agile Risk Management LLC Nigilant32 Active Memory Imaging Beta v 0.1: http://www.agileriskmanagement.com/publications_4.html

Michael, C., & David, C. (2007, May 27). *PyFlag Forensic And Log Analysis GUI*. Retrieved July 11, 2010, from PyFlag: http://pyflag.sourceforge.net/Presentations/PyFlag_Auscert2007/

Michael, L., Julia, A., & Marc, R. (2006, February). Gap Analysis: Judicial Experience and Perception of Electronic Evidence. *Journal of Digital Forensic Practice, I*(1), 13-17.

Microsoft. (2007, August 7). *Microsoft Corporation*. Retrieved July 20, 2011, from UNICODE:Adobe Encoding Character Code - Search supports 8-bit (UTF-8): http://support.dtsearch.com/dts0140.htm

Microsoft. (2007, August 10). *Microsoft Corporations*. Retrieved March 23, 2009, from Windows XP: http://windows.microsoft.com/en-US/windows/products/windows-xp

Microsoft. (2008, Accessed August). *Microsoft Corporation*. Retrieved from Microsoft Windows Vista: http://windows.microsoft.com/en-US/windows-vista/products/home

Microsoft. (2008, March 1). *Microsoft Corporation*. Retrieved April 15, 2011, from MSDN - UNICODE: http://msdn.microsoft.com/en-us/library/dd374081(v=vs.85).aspx

Microsoft. (2011, Accessed September). *Microsoft Corporation*. Retrieved from Microsoft Windows 7: http://windows.microsoft.com/en-US/windows7/products/home

Msuiche. (2008, March 2). *Capture memory under win2k3 or vista with win32dd.* Retrieved July 11, 2009, from Msuiche.net: http://www.msuiche.net/2008/06/14/capture-memory-under-win2k3-orvista-with-win32dd.

Nicolas, R. (2008, October 10). Windows memory forensic. *Journal in Computer Virology, 4*(2), 83-100.

Nicole, L. B., & Jan, G. C. (2005). A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation, 2*(2), 146-166.

NIJ. (2001, July 1). *National Institute of Justice.* Retrieved May 18, 2011, from Electronic Crime Scene Investigation a Guide for First Responders: http://www.ncjrs.org/pdffiles1/nij/187736.pdf.

NIJ. (2008, April 1). *National Institute of Justice.* Retrieved October 12, 2010, from Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition. www.ojp.usdoj.gov/nij: http://www.nwfia.org/NIJGuideforFirstResponders.pdf

NIST. (2008, January 24). *Computer Forensics Tool Testing (CFTT) Project:Forensic String Searching Tool Requirements Specification.* Retrieved December 23, 2010, from National Institute of Standard Technology: http://www.cftt.nist.gov/ss-req-sc-draft-v1_0.pdf

NIST. (2009, March 1). Retrieved June 20, 2011, from National Institute of Standard Technology - Forensic String Search Specification, V1, Draft 1. March 2009.: http://www.cftt.nist.gov/StringSearch.html;

NIST. (2010, July 29). *National Institute of Standards and Technology: Container Files- String searching on container and nested container files.* Retrieved April 5, 2011, from The CFReDS Project: Computer Forensic Reference Data Sets (CFReDS) for digital evidence: http://www.cfreds.nist.gov/SearchingContainerFiles.html

Okolica, J., & Gilbert, L. P. (2011). Extracting the Windows clipboard from physical memory. *Digital Investigation, 8*(14), 118-124.

Orin, S. K. (2005, February). Digital evidence and the New Criminal Procedure. *Columbia Law Review, 105*(1), 279-318.

Oxford, C. (2011, March 21). *Oxford Dictionaries*. Retrieved July 28, 2011, from The OEC: Facts About the Language: http://oxforddictionaries.com/page/oecfactslanguage/the-oec-facts-about-the-language

Palmer, G. (2001). *A Road Map for Digital Forensic Research.* The First Digital Forensic Research Workshop (DFRWS)). Utica, NY: Digital Forensic Research Workshop .

Pavel, G. (2004, June 30). *Formalising Event Reconstruction in Digital Investigations.* Retrieved August 21, 2010, from A Formal Forensic Organisation: http://www.formalforensics.org/publications/thesis/index.html

Peter, J. (2011, March 7). Retrieved July 14, 2011, from BBC NEWS Magazine:100 Words of English: How Far Can It Get You: http://www.bbc.co.uk/news/magazine-12894638

Peter, S. (2009). *Forensic Science Standards in Fast-Changing Environments.* Testing Methodologies in Digital Forensics given at EAFS 2009, London School of Economics & Political Science, Open University, UK, London.

Peter, S. (2011, June). Certification, Registration and Assessment of Digital Forensic Experts: The UK Experience. *Digital Investigation, VI*(1), 1-8.

Peterson, G., & Shenoi, S. (2009). *Advances in Digital Forensics V - Fifth IFIP WG 11.9 International Conference on Digital Forensics* (5th Edition ed.). (S. S. Peterson Gilbert, Ed.) Orlando, Florida,, USA: Springer.

Pollitt, M. (1995). *Computer Forensics: an approach to evidence in cyberspace.* Baltimore, MD.: National Information Systems Security Conference.

R.B., van Baar, W., Alink, A. v., & Ballegooij. (2008). Forensic memory analysis: Files mapped in memory. *Digital Investigation, 5*(14), 52-57.

Richard, M. S., & Eoghan, C. (2010). Extracting Windows command line details from physical memory. *Digital Investigation, 7*(8), 57-63.

Richard, N., Colin, O., Jake, B., & Cal, W. (2005). *First Responders Guide to Computer Forensics.* Handbook, United States Department of Justice Cyber Crime:CERT Training and Education, Hanscom AFB, MA 01731-2100.

Rolf, O., & Ruedi, R. (2003, Septenber). Digital evidence: Dream and Reality. *IEEE Security & Privacy, 3*(5), 44-48.

Ruibin, G., Tony, K. Y., & Mathias, G. (2005, April). Case-relevance information investigation: binding computer intelligence to current computer forensic framework. *International Journal of Digital Evidence, II*(2), 147-167.

Russinovich, M., & Solomon, D. A. (2009). *Microsoft Windows internal covering Windows Server 2008 and Windows Vista* (5th ed.). Washington, USA: Microsoft Press.

Schuster, A. (2006). Searching for processes and threads in microsoft windows memory dumps. *Digital Forensic Research Workshop (DFRWS),, 3*(6), 10-16.

Siti, R. S., Robiah, Y., & Shahrin, S. (2008). Mapping Process of Digital Forensic Investigation Framework. *International Journal of Computer Science and Network Security (IJCSNS), 8*(10), 163-169.

Stefan, K., Tobias, H., & Jana, D. (2009). A new forensic model and its application to the collection, extraction and long term of screen content off a memory dump. *16th International Conference on Digital Signal Processing (DSP)* (pp. 23-89). Aegean island of Santorini, Greece: Digital Signal Proceeding.

Stephen, M. (2008, September 1). Judges and Technical Evidence. Keynote address at CFET 2008:The 2nd International Conference on Cyberforensics Education and Training, Canterbury Christ Church University. Canterbury, Canterbury, UK.

Stephenson, P. (2003, June). A comprehensive approach to digital incident investigation. *Information Security Technical Report, 8*(2), 42-54.

Stuart, J., & Jon, J. N. (2005). *Forensic Science: An introduction to scientific and investigative techniques* (2nd ed.). Boca Raton, Florida: Taylor & Francis Group CRC Press.

Stuart, J., Jon, J. N., & Suzanne, B. (2007, May). *Forensic science introduction to scientific and investigative techniques* (3rd Edition ed., Vol. II). (J. J. Stuart James, Ed.) Boca Raton, Fla, Fla: CRC.

SWGDE. (2010, July 5). Retrieved November 29, 2011, from Scientific Working Group on Digital Evidence - Recommended Guidelines for Validation Testing Version: www.swgde.org;

Timothy, V. (2007, October). The acquisition and analysis of random access memory. *Journal of Digital Forensic Practice, 1*(4), 315-323.

Todd, G. S., & Henry, R. R. (2006). *Collecting Evidence from a running computer:A Technical and Legal Primer for the Justice Community.* Denver, Colorado: SEARCH, The National Consortium for Justice Information and Statistics.

US-CERT. (2008, June 2). Retrieved September 14, 2011, from US-CERT Computer Forensic:A government organization: http://www.us-cert.gov/reading_room/forensics.pdf

Walters, A., & Petroni, N. (2007, February). Volatools: Integrating volatile memory into the digital investigation process. *Black Hat DC 2007*, 1-18.

Warren, R. G. (2008, March). Empirical Studies of Query and Document Characteristics as Evidence in Favor of Relevance. *International Journal of Legal Evidence, II*(2), 1-9.

APPENDICES

Appendix A

List of Publications

Conference publications most relevant to this thesis

1. F. Olajide, N. Savage, “Extraction Of User Information By Pattern Matching Techniques In Windows Physical Memory”, The International Conference on Digital Enterprise And Information Systems (DEIS2011), London Metropolitan University, London, UK. 20-22 July 2011.
2. F. Olajide, N. Savage, “Forensic Extraction of User Information in Continuous Block of Evidence”, International Conference on Information Society (i-Society2011), London, UK. 27-29 June 2011, London, UK.
3. F. Olajide, N. Savage, “Digital Forensic Research and Method of Extracting Relevant Information From Physical Memory Of Windows Systems”, Fourth International Conference on Internet Technologies and Applications (ITA 11), Glyndwr University, North Wales, Wrexham, UK. 6-9 September 2011.
4. F. Olajide, N. Savage, “Dispersal of Time Sensitive Evidence in Windows Physical Memory”, Cyberforensics 2011, International Conference on Cybercrime, Security & Digital Forensic, The University of Strathclyde, Glasgow, UK, 27 – 29 June 2011
5. F. Olajide, N. Savage, "On the Extraction Of Forensically Relevant Information From Physical Memory", WORLDCIS, Computer Internet Security, UK, 21–23 Feb 2011
6. F. Olajide, N. Savage, D. Ndzi, H. Al-Sinani, "Forensic Live Response and Event Reconstruction", PG-Net2009 Conference, Convergence of Telecommunications, Networking & Broadcasting, Liverpool, U.K., 22-23 June 2009

Journal publications most relevant to this thesis

7. F. Olajide, N. Savage, "On the Identification Of Information Extracted From Windows Physical Memory", International Journal for Information Security Research (IJISR), Volume 1, Issue 3, ISSN: 2042-4639., 2011.
8. F. Olajide, N. Savage, "Application Level Evidence and Event Reconstruction", Journal of Computing in Systems- ISSN 1472-9083 (Source Code – GO, Country Code 2 - Shelf mark 4964.1500000, U.K., Dec., 2010.

Appendix B

Sample MS Excel in-built system defined data that resided in the application memory

```
#################################################################
#                                                               #
#    Sample existence textual in-built system defined data stored in the        #
#    MS EXCEL application memory. This list of existed data was extracted  #
#    from the memory dump of the application                    #
#################################################################
```

118
FSCheckbox
119
FSButtonLabel
120
FSEnterStringLabel
121
FSLabel
122
ExecuteThisOrOtherAction
123
scrollbutton
124
GalExpandButton
125
FSFlyoutAnchor
126
gbutton
127
StatusLabel
128
StatusLabelDisabled
103
English (United Kingdom)
104
English (United Kingdom)
105
English (United States)
129
TabLabels
130
TabLabel
131
Standard
972
973
ShowLabel
974
Commands
975
showLabel
976
ShowLabel
977
showImage
978
getLabel
979
checkBox
980
Commands
981
RTFLabel
982
RibbonTabLabel
992
Sheet_Title
993
Commands
994
Check Box
995
Label
996
&Group and Outline
997
S&heet
998
S&heet
999
&Sheet
1000
Grand Count
1003
Grand Min
1002
Grand Max
1001
Grand Average
1004
Grand Product
1005
Grand StdDev
1006
Grand StdDevp
1007
Grand Var
1008
Grand Varp
1009
Grand Total
1010
Row Grand Total
1011
Column Grand Total
1012
Count Numbers
1013
CatTickLabel
1014
DispUnitLabel
1016
Label
1017
B3IsCheckbox
1018
B3Layouts

commands
1468
label
1469
labelIntro
1470
labelColRow
1471
button
1472
Label
1474
labelWith
1475
checkBox
1550
N&umbers
1551
&Landscape
1552
Fi&rst page number:
1553
Landscape sample
1554
&Different odd and even pages
1555
&Black and white
1556
Row and co&lumn headings
1557
&Down, then over
1558
O&ver, then down
1559
Sheet sample
1560
Print in &black and white
1561
Fi&nd what:
1562
Match &Kashidas
1563
Help on this function
1566
Create names from values in the:
1568
1567
Series and &data point
1570
&Radar axis labels
1571
Series distribution
1572
Calc &Sheet
1573
Sheet options
1574
Recalc&ulate before save
1575
Refresh control
1576
&Refresh every
1600
No header ro&w
1601
All using Source t&heme
1602
Fo&rmulas and number formats
1603
Val&ues and number formats
1628
IsDistributed
1629
LargeLayout
1630
MediumLayout
1631
SmallLayout
1787
simplebutton
1788
button
1789
IsChecked
1790
label
1791
checkbox
1792
radiobutton
1793
Repeatbutton
1794
isThemeEnabled
1795
isThemeActive
1796
label
1797
label
1798
label
1799
button
1800
label
1801
button
1802
label
1803
listLabel
1804
textLabel
1805
button
1806
label
1807
listLabel
1808
button
1809
label
1810
checkbox
1811
button
1812
label
1813
ListLayout
1814
button

&Series number:
1815
label
1569
&Point number:
1816
button
1817
label
1818
button
1819
checkBox
1820
label
1821
idSiteSelectLabel
1822
expandToFillHoriz
1823
associatedLabelID
1824
label
1825
ListLayout
1826
button
1827
Groupheader
1828
checkbox
1829
label
1830
button
1831
label
1832
ListLayout
1833
button
1834
checkBox
1835
label
1836
1839
searchbutton
1840
isdefaultbutton
1841
layoutposition
1842
expandtofillhoriz
1843
helplink
1844
awsgroupheader
1845
height
1846
helpid
1847
label
1848
groupheader
1849
searchbutton
1850
isdefaultbutton
1851
layoutposition
1909
Label
1910
titleLabel
1911
statusLabel
1912
priorityLabel
1913
assignedToLabel
1914
descriptionLabel
1915
duedateLabel
1916
expandToFillHoriz
1917
associatedLabelID
1918
Label
1919
fileLabel
1920
Checkbox
1921
Label
1922
idFolderLabel
1923
Label
1924
urlLabel
1925
descriptionLabel
1926
commentsLabel
1927
Label
1928
ItemLabel
1929
button
1930
Label
1931
idLabel
1932
Label
1933
button
1934
IsChecked
1935
checkbox
1936
radiobutton
1937
label
1938
TreeViewContentLabel
1939
helplink
1940
awsgroupheader
1941
searchbutton

button
1837
label
1838
simplebutton
1942
NumberOfLinks
1943
label
1944
button
1945
groupheader
1946
label
2982
VerticalNotExpanded
2983
groupheader
2984
Label
2985
BoxLayout
2986
conceptLabel
2987
HideLabel
2988
SimpleHorizontalDistributed
2989
VerticalNotExpanded
2990
groupheader
2991
Label
2992
Number
2993
SimpleHorizontalDistributed
2994
VerticalNotExpanded
2995
groupheader
2996
Label
2997
3000
FlyoutAnchor
3001
Standard
3002
Label
3003
ConfigLabel
3004
ConfigLabel
3005
Label
3006
StandardItems
3007
Theme
3008
Label
3009
BoundLabel
3010
button
3011
groupheader
3012
InfoTextLabel
3013
label
3014
button
3015
CustomColorSchemeDlg
3016
label
3017
conceptLabel
3018
descLabel
3019
groupheader
3020
HorizontalLayout
3021
HorizontalLayoutAlignRight
3022
button
3023
#CustomColorSchemeDlg
3024
DeleteScheme
3025
ResetScheme
3026
PreviewScheme
3027
OartColorSchemeDlgParam
s
3028
DeleteScheme
3029
ResetScheme
3030
PreviewScheme
3031
CustomFontSchemeDlg
3032
label
3033
conceptLabel
3034
descLabel
3035
groupheader
3073
Label
3074
LabelPosition
3075
Label
3076
NumberFormat
3077
NumberFormatGeneral
3078
NumberFormatNumber
3079
NumberFormatCurrency
3080
numberFormatStringList

NumbersGallery
2998
Standard
2999
Label
3081
Label
3082
NumberFormatAccounting
3083
NumberFormatDate
3084
NumberFormatTime
3085
NumberFormatPercentage
3086
NumberFormatFraction
3087
NumberFormatScientific
3088
NumberFormatText
3089
NumberFormatSpecial
3090
NumberFormatCustom
3091
LabelCrossesAt
3092
IndexTickMarkLabels
3093
ValueShowDisplayLabel
3094
LabelDisplayUnits0
3095
LabelDisplayUnits1
3096
LabelDisplayUnits2
3097
LabelDisplayUnits3
33255
BuildCheckSum
33256
BuildCheckSum
33257
BuildCheckSum
33258
33261
BuildCheckSum
33262
BuildCheckSum
33263
BuildCheckSum
33264
BuildCheckSum
33265
BuildCheckSum
33266
BuildCheckSum
33267
Publisher
33268
BuildCheckSum
33269
BuildCheckSum
33270
BuildCheckSum
33271
BuildCheckSum
33272
BuildCheckSum
33273
BuildCheckSum
33274
BuildCheckSum
33275
BuildCheckSum
33276
BuildCheckSum
33277
BuildCheckSum
33278
BuildCheckSum
33279
BuildCheckSum
33280
BuildCheckSum
33281
BuildCheckSum
33282
BuildCheckSum
33283
BuildCheckSum
33284
BuildCheckSum
33285
BuildCheckSum
33286
BuildCheckSum
33287
BuildCheckSum
33288
BuildCheckSum
33289
BuildCheckSum
33290
BuildCheckSum
33291
BuildCheckSum
33292
BuildCheckSum
33293
BuildCheckSum
33294
BuildCheckSum
33295
BuildCheckSum
33296
BuildCheckSum
33297
BuildCheckSum
33298
BuildCheckSum
33299
BuildCheckSum
33300
BuildCheckSum
33301
BuildCheckSum
33302
BuildCheckSum
33303
BuildCheckSum
33304
Publisher
33305
BuildCheckSum
33306
BuildCheckSum

BuildCheckSum
33259
BuildCheckSum
33260
BuildCheckSum
33307
BuildCheckSum
33308
BuildCheckSum
33309
BuildCheckSum
33310
BuildCheckSum
33311
Publisher
33312
BuildCheckSum
33313
BuildCheckSum
33314
BuildCheckSum
33315
BuildCheckSum
33316
BuildCheckSum
33317
BuildCheckSum
33318
BuildCheckSum
33319
BuildCheckSum
33320
BuildCheckSum
33321
BuildCheckSum

Appendix C

Sample MS Access in-built system defined data that resided in the application memory

```
#########################################################
#                                                       #
#    Sample existence in-built system defined data stored in the        #
#   MS Access application memory. This list of existed data was      #
#   extracted from the memory dump of the application                  #
#                                                       #
#########################################################
```

41
English (United Kingdom)
42
TreeViewContentLabel
43
CategoryHeaderLabel
44
GalleryItemLabelContainer
45
FSCheckbox
46
FSButtonLabel
47
FSEnterStringLabel
48
FSLabel
49
ExecuteThisOrOtherAction
50
scrollbutton
51
GalExpandButton
52
FSFlyoutAnchor
53
gbutton
54
StatusLabel
55
StatusLabelDisabled
56
TabLabels
57
TabLabel
58
Standard
59
StandardItems
60
ListColumnLabel
61
HideLabel
62
CheckLocallyHidden
63
conceptCheck
64
descLabel
65
conceptLabel
66
LabelAbove
67
DropLabel
68
LayoutElement
69
VertLayout
70
highlightoverlayouter
71
ThemeItemOverlayOuter
72
ThemeItemOverlayInner
73
GA_LabelOutside
74
GA_Label
75
SB_Label
76
DD_Label
77
FSDD_LabelArrow
78
CB_Label
85
AcceleratorLabel
86
valueLabel
87
LongLabel
88
CommandItemList
89
CommandItem
90
HorizontalGroupExpanded
91
RowExpanded
102
GroupedPathLabel
103
LabelOnly
104
headerHolder
105
mainLabel
106
catButtonLabel

107
spinnerLabel
108
BrandingSpace
862
Attachments_Label
863
Opened Date_Label
864
Opened_Date_Label
865
cmdContactList_LayoutLabel
866
cboReports_Label_LayoutLabel
867
cboReports_LayoutLabel
986
DataSource.Layout
987
DataSource.Label
988
DataSource.Commands
989
SelectedItem.Label
990
SelectedItem.Label
991
DataContext.Label
1074
label
1075
label
1076
label
1077
button
1078
label
1079
button
1080
label
1081
listLabel
1082
textLabel
1083
button
1084
label
1085
listLabel
1086
button
1087
label
1088
checkbox
1089
button
1090
label
1091
ListLayout
1092
button
1093
label
1094
button
1095
label
1096
button
1097
checkBox
1098
label
1099
idSiteSelectLabel
1100
expandToFillHoriz
1101
associatedLabelID
1102
label
1103
ListLayout
1104
button
1105
Groupheader
1106
checkbox
1107
label
1108
button
1109
label
1110
ListLayout
1111
button
1112
checkBox
1113
label
1114
button
1115
label
1116
simplebutton
1117
searchbutton
1118
isdefaultbutton
1119
layoutposition
1120
expandtofillhoriz
1121
helplink
1122
awsgroupheader
1123
height
1124
helpid
1125
label
1126
groupheader
1127
searchbutton
1128
isdefaultbutton

1129
layoutposition
1130
expandtofillhoriz
1131
helplink
1132
awsgroupheader
1133
height
1134
helpid
1135
label
1136
simplebutton
1137
groupheader
1187
Label
1188
titleLabel
1189
statusLabel
1190
priorityLabel
1191
assignedToLabel
1192
descriptionLabel
1197
fileLabel
1198
Checkbox
1199
Label
1200
idFolderLabel
1201
Label
1202
urlLabel
1203
descriptionLabel
1204
commentsLabel
1205
Label
1206
ItemLabel
1207
button
1208
Label
1209
idLabel
1210
Label
1211
button
1212
IsChecked
1213
checkbox
1214
radiobutton
1215
label
33849
CustomLabels
33850
CustomLabel
33851
LineNumbering
33852
CaptionLabels
33853
CaptionLabel
33854
PageNumbers
33855
PageNumber
33856
HangulAndAlphabetExceptions
33857
HangulAndAlphabetException
33858
OtherCorrectionsExceptions
33859
OtherCorrectionsException
33860
StyleSheet
33861
StyleSheets
33862
XMLSchemaReference
33863
XMLSchemaReferences

Appendix D

Sample qualitative assessment of user input activities that has been reconstructed as extracted on the application and rearranged to form sentences of words in whole fragment and phrase of words in partial fragment of information.

```
##################################################
#                                                #
#  Sample Partial Fragment of user input that was rearranged  #
#                                                #
##################################################
```

64
test1-United top world rich list despite
65
Day 1- 22042010
80
C:\Documents and Settings\Administrator\My Documents\PHD TASK 3\Day 1- 22042010\test1-United top world rich list despite
132146
My Documents\PHD TASK 3\Day 1 - 23042010\test1-Lib Dems plan new house tax.docx
132139
My Documents\PHD TASK 3\Day 1 - 23042010\test4-the flowerpothole man.docx
132150
My Documents\PHD TASK 3\Day 1 - 23042010\test3-Pompey lay bare extent if financial.docx
132147
\test1-Lib Dems plan new house tax.docx\
132242
Microsoft Office Word
59772
FPC
132217
Microsoft Office Word
68063
Word is preparing to background save test1-United top world rich list despite:
59772
United top world rich list despite
59776
700m debt

##
#   #
Sample Partial Fragment of user input that was rearranged
#   #
##

Fummy Prolite Computer
5639
132241
WINWORD.EXE
410
top world rich list despite.docx
132219
Save a copy of the document that is fully compatible with Word 97-2003.

###
#   #
Sample Partial Fragment of user input that was rearranged
#   #
###

5516
C:\Program Files\Microsoft Office\Office12\WINWORD.EXE
421
C:\Documents and Settings\Administrator\My Documents\PHD TASK 3\Day 1- 22042010\test1-United top world rich list despite.docx
513
United top world rich list despite.docx
1586
test1-United top world rich list despite - Microsoft Word
7116
C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\MY DOCUMENTS\PHD TASK 3\DAY 1- 22042010\TEST1-UNITED TOP WORLD RICH LIST DESPITE.DOCX
59770
Uni Port
59771
Fummy Prolite Computer
5534
HP Deskjet 6940 series
5823
C:\Program Files\Microsoft Office\Office12\WINWORD.EXE
5513
Windows NT x86
2267137
c:\Documents and Settings\All Users\Start Menu\Programs\Microsoft Office\Microsoft Office Tools\Microsoft Office 2007 Language Settings.lnk

2275936
Microsoft Office Word 2007.lnk
132213
Word is saving test1-United top world rich list despite: (100%, Word is preparing to background save test1-United top world rich list despite:
44693
United top world rich list despite.docx

##
#   #
Sample Whole Fragment of user input that was rearranged
#   #
##

59764
MANCHESTER United have been valued as the biggest football club in the world. The Old Trafford side, who are more than 700million in debt, held on top spot in Forbes Magazine's list of the world's 20 most valuable football teams. Six of that top 20 are from England - despite the Premier League being the most indebted in Europe, according to governing body Uefa.

##
#   #
Sample Partial Fragment of user input that was rearranged
#   #
##

59772
United top world rich list despite
59776
700m debt

##
#   #
Sample Whole Fragment of user input that was rearranged
#   #
##

59765
MANCHESTER United have been valued as the biggest football club in the world.

##
#   #
Sample Partial Fragment of user input that was rearranged
#   #
##

59766
The Old Trafford side, who are more than
59767
700million in debt, held on top spot in Forbes Magazine'
77806
s list of the world
348436
of the world's 20 most valuablLeague being the most indebted in Europe, according to governing body Uefa.
353726
according to

##
#   #
Sample Whole Fragment of user input that was rearranged
#   #
##

59758
"s 20 most valuable football teams. Six of that top 20 are from England - despite "the Premier League being the most indebted in Europe, according to governing body Uefa."

##
#   #
Sample Whole Fragment of user input that was rearranged
#   #
##

77806
 despite the Premier League being the most indebted in Europe, according to governing body Uefa.</w:t></w:r></w:p><w:sectPr w:rsidR="006F03F8" w:rsidRPr="006F03F8" w:rsidSect="00A803D6"><w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="708" w:footer="708" w:gutter="0"/><w:cols w:space="708"/><w:docGrid w:linePitch="360"/></w:sectPr></w:body></w:document>
59759
"the Premier League being the most indebted in Europe, according to governing body Uefa."

##
#   #
Sample Partial Fragment of user input that was rearranged
#   #
##

77809
United top world rich list despite
59777
the Premier League being the most indebted in Europe, according to governing body Uefa.s Magazine'
135787
according to

##
#   #
Sample Partial Fragment of user input that was rearranged
#   #
##

101515
the Premier L
146491
eague being the most indebted in Europe, according to governing body Uefa.g body Uefa.s Magazine'
101521
League being the most indebted in Europe, according to governing body Uefa.re i Fofrom England