Anti-Forensic Implications of Software Bugs in Digital Forensic Tools

ALAIN HOMEWOOD
Bachelor of Computer and Information Sciences (AUT, NZ)
Diploma in Information Technology (AUT, NZ)

A thesis submitted to the graduate faculty of design and creative technologies
AUT University
in partial fulfilment of the
requirements for the degree of
Master of Forensic Information Technology

School of Computing and Mathematical Sciences

Auckland, New Zealand
2012

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a University or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.

.....

Signature

Acknowledgements

I would like to thank everyone that has supported me in writing this thesis. Without their support this thesis may not have been completed. I have received guidance, support and insight from many people during the course of conducting this research.

Firstly I would like to express my gratitude to my thesis supervisor, Dr Brian Cusack. Dr Cusack has provided continual support and guidance which has greatly influenced the direction of this research as well as the quality of the thesis. Dr Cusack is the latest of many faculty members of the School of Computing and Mathematical Sciences who have helped me reach my goals throughout my tertiary education.

I also thank Campbell McKenzie for his continued support throughout writing my thesis. Campbell has provided me with access to many resources as well as a flexible working arrangement, without which this research would not have been possible. Thanks must also be extended to my colleagues Ben Knight and Ian Donovan who have been supportive of my flexible working arrangement.

Lastly, I would like to thank all of my fellow students from the Master of Forensic Information Technology. Many of whom have provided continued friendship and inspiration throughout my studies.

Abstract

The digital forensic community relies on a small number of complex tools to analyse digital evidence. These digital forensic tools have greatly improved the accuracy and efficiency of investigations. However, the reliance on tools may be a weakness that can be exploited to prevent or disrupt investigations. Counter-measures to digital forensic techniques, known as anti-forensics, have typically been focussed on techniques to hide or prevent the creation of evidence. The concern of the author is that anti-forensic techniques may soon be focussed on exploiting software bugs in digital forensic tools. The tools used by the digital forensic community are complex with many different functions, which may contain software bugs. The risk of such software bugs is that digital forensic investigations could be compromised. This research evaluates the potential anti-forensic risk and implications of software bugs in digital forensic tools.

This research first presents a literature review of areas of digital forensics related to anti-forensic risk such as anti-forensic techniques, tool testing methodologies and legal issues. This research then develops a suitable methodology to identify software bugs in digital forensic tools with potential anti-forensic risk. The methodology consists of six test cases designed to test various function areas of digital forensic tools for the presence of software bugs. Each test case has associated with it a number of reference sets to be used as input, which contain deliberately malformed data created through the process of file fuzzing. Acceptance spectrums ranging from “critically unacceptable” to “exceeds expectations” were developed to evaluate the anti-forensic risk caused by the identified software bugs.

The research was successful in identifying a number of software bugs, the majority of which resulted in the digital forensic tools crashing. The software bugs identified were evaluated for anti-forensic risk and four test cases were determined to pose an unacceptable anti-forensic risk. Two test cases were determined to exceed expectations due to no software bugs being identified.

The conclusion of the research is that software bugs in complex function areas of digital forensic tools pose an unacceptable anti-forensic risk. No critically unacceptable risks could be identified by this research. There is potential for further research into the anti-forensic implications of such software bugs.

Table of Contents

Chapter One – Introduction

Chapter Two – Literature Review

Chapter Three – Research Methodology

Chapter Four – Research Findings

Chapter Five – Discussion

Chapter Six – Conclusion

Appendix

List of Tables

List of Figures

List of Abbreviations

ASLR	Address Space Layout Randomisation
BIOS	Basic Input Output System
CEIC	Computer and Enterprise Investigations Conference
CFTT	Computer Forensic Tool Testing Program
CPU	Central Processing Unit
DCO	Device Configuration Overlay
DEP	Data Execution Prevention
EC	EnCase
EP	Evidence Processor
ESATA	External Serial Advanced Technology Attachment
FN	Filename
FTK	Forensic Toolkit
GB	Gigabyte
GPT	Globally Unique Identifier Partition Table
HDD	Hard Disk Drive
HPA	Host Protected Area
IBM	International Business Machines Corporation
IEC	International Electrotechnical Commission
IM	Instant Message
IPS	Intrusion Prevention System
ISO	International Organisation for Standardization
IT	Information Technology
JPEG	Joint Photographic Experts Group
MACE	Modified, Accessed, Created and Entry modified
MBR	Master Boot Record
MFT	Master File Table
MRU	Most Recently Used
NIJ	National Institute of Justice
NIST	National Institute of Standards and Technology
NTFS	New Technology File System
PDF	Portable Document Format
RAM	Random Access Memory

RS	Reference Set
SANS	SysAdmin, Audit, Networking and Security
SATA	Serial Advanced Technology Attachment
SIA	Standard Information Attribute
SLR	Scalable Linear Recording
SSD	Solid State Drive
SWGDE	Scientific Working Group on Digital Evidence
TB	Terabyte
TC	Test Case
TSK	The Sleuth Kit
US	United States
USA	United States of America
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UTF-8	Universal Character Set Transformation Format - 8 Bit

Chapter One

INTRODUCTION

1.0 BACKGROUND

Law enforcement agencies in the United States began working together in dedicated units to combat computer-related crimes in the late 1980's (Casey, 2011, p. 10). However the early pioneers of digital forensics did not have any dedicated forensic tools and often investigations were performed as live analysis using the suspect device to view evidence. As the field of digital forensics evolved, specialised tools were developed to assist with digital forensic investigations. The early digital forensic tools were the first attempts at safe and efficient collection of evidence. Once the evidence was collected, the analysis phase was still very much a tedious manual task. As the volume and types of evidence encountered by digital forensic practitioners increased, it became more apparent that manual forensic analysis was inefficient and impractical. More advanced forensic tools soon appeared that performed and automated complex analysis tasks.

Advanced digital forensic tools such as EnCase Forensic ("EnCase") and Forensic Toolkit (FTK) have become a de facto standard in the digital forensics community. A tool such as EnCase is accepted and relied upon by digital forensic practitioners and courtrooms around the world. Digital forensic practitioners commonly rely on tools to perform complex forensic analysis processes. The reliance on digital forensic tools has resulted in a situation where a digital forensic practitioner only requires a minimal level of knowledge about how a tool works to analyse digital evidence. To ensure the integrity of evidence produced by tools, organisations such as the National Institute of Standards and Technology (NIST) have performed testing of many of the common function areas of tools such as acquisition and keyword searching. However, some in the forensic community such as Sommer (2010) are concerned that digital forensic tools are not being held to the same standards as other forensic sciences. Others, such as Carrier (2002), advocate the use of open source digital forensic tools to allow for greater scrutiny of how digital forensic tools work.

The widespread adoption and use of fingerprinting techniques by law enforcement agencies compelled criminals to develop counter-measures such as wearing gloves. Similarly, in digital forensics, a number of counter-measures to prevent or disrupt forensic analysis have been developed and are commonly referred to as anti-forensics. A suspect who knows their computer is about to be seized and copied for forensic analysis might use a wiping program to destroy evidence or hide evidence in an encrypted container. Simple anti-forensic techniques such as wiping a drive are not uncommon in digital forensic investigations. The use of more complex anti-forensic techniques, such as creating counterfeit evidence has been discussed by researchers such as Harris (2006). Carrier (2002) acknowledges that complex anti-forensic techniques could be used but suggests that anti-forensic techniques would not completely prevent an investigation.

Currently the digital forensic community depends on a small set of partially-tested tools from a handful of vendors. The tools being used have had some function areas tested and verified by organisations such as NIST but for most function areas, the vendor is relied on to have performed sufficient testing. Researchers such as Rogers (2005, p. 9) suggest that dependence on a small number of tools is making the forensic community vulnerable to anti-forensic risk. A software bug that could exist in EnCase or FTK has the potential to affect a large portion of the digital forensic community. If the software bug presents a vulnerability that creates an anti-forensic risk then there is the possibility of wide spread disruption of digital forensic investigations.

The aim of this research is to investigate the extent that software bugs in digital forensic tools make digital forensic tools vulnerable to anti-forensic risk. The research presents a number of test cases where deliberately malformed data is input into a digital forensic tool in an attempt to locate software bugs. The software bugs identified are then analysed and a discussion of the associated anti-forensic risk is presented. The purpose of the research is to provide a practical demonstration of software bug identification in digital forensic tools and the associated anti-forensic risk. The research question to be addressed by the research is:

What is the anti-forensic risk caused by software bugs in digital forensic tools?

1.1 MOTIVATION

A literature review reveals a number of examples of flaws in digital forensic tools as well as examples of demonstrated tool-related anti-forensic risks. The risks range from Ayers' (2009) discovery of several flaws in the way EnCase handles dates and times to Neckar and Ose (2010, pp. 27-34) who demonstrated a code execution vulnerability that affected both EnCase and FTK. The digital forensic community is aware that digital forensic tools have had issues in the past but the consequences have been minimal. When each issue has been identified what results is usually a minor discussion by members of the digital forensic community and a vendor response to minimise concern. At this stage the author has found no documented evidence of any software bug in a digital forensic tool resulting in a severe anti-forensic risk that has been exploited maliciously to prevent or disrupt an investigation.

When the forensic community were alerted to the code execution vulnerability in EnCase by McCash (2010) it was met with a small response suggesting the digital forensic community may be complacent and unconcerned with regards to anti-forensic risk. Complacency is perhaps justified in that even if a severe anti-forensic risk could be proven to exist in a digital forensic tool there would likely be little impact on the day-to-day work of a digital forensic practitioner.

The concern of the author is that in the near future anti-forensic techniques and exploitation of software bugs will present a much more credible and serious threat than currently. It is not uncommon for creators of malicious software to use sophisticated counter-measures against anti-malware software and reverse engineering tools. Similarly it is not difficult to envisage a criminal organisation or state sponsored organisation using sophisticated anti-forensic techniques to prevent or disrupt the use of digital forensic tools. Digital forensic practitioners are used to with dealing with a suspect who tries to wipe their hard drive when the police knock on their door. However, digital forensic practitioners are not prepared to deal with a well-resourced organisation that can perform their own research and development of anti-forensic techniques.

The motivation for conducting this research is to provide a better understanding of the anti-forensic risks associated with software bugs in digital

forensic tools. Previous research from the digital forensic field has largely focused on theoretical discussions of potential anti-forensic risk and potential counter-measures. This research aims to provide a more practical demonstration of software bugs in digital forensic tools and a discussion of observable anti-forensic risk. The author believes that practical experimentation and testing is needed to provide some tangible evidence and insight regarding the extent of anti-forensic risk in digital forensic tools. There are two primary benefits of practical testing as related to anti-forensic risk. Firstly, the digital forensic community is able to identify and counter anti-forensic risk before someone does so maliciously. Secondly, digital forensic practitioners and courtrooms can gain a greater level of confidence in the tools used even if no anti-forensic risk is identified.

1.2 STRUCTURE OF THESIS

The thesis consists of six chapters: Chapter One “Introduction”, Chapter Two “Literature Review”, Chapter Three “Research Methodology”, Chapter Four “Research Findings”, Chapter Five “Discussion” and Chapter Six “Conclusion”.

Chapter Two presents a review of literature in areas of digital forensics relevant to anti-forensic risk. Areas reviewed by Chapter Two include the history of digital forensic tools, anti-forensic techniques and risks, evidential and legal issues and tool-testing methodologies. Chapter Two also presents a case study of anti-forensic risks associated with the digital forensic tool EnCase. Chapter Two is split into five main Sections: Digital Forensic Tools, Anti-Forensic Risk, EnCase Case Study, Evidential Implications and Tool Risk Evaluation. The literature reviewed by Chapter Two provides context and background for the research presented in this thesis, and provides an overview of the current state of research into the field of anti-forensic risk.

Chapter Three builds on the literature reviewed in Chapter Two to develop a methodology for testing digital forensic tools for software bugs and the associated anti-forensic risk. Chapter Three begins by first reviewing five similar studies in areas including currently used tool testing methodologies and alternative tool testing methodologies. The review of similar studies is then followed by a review of the problem areas identified in the literature. A research question and sub-questions are then derived from this review. A series of

hypotheses for the research question are also developed which will be tested following the collection of field findings. Chapter Three then presents a research methodology consisting of four phases, specifying how the field testing is to be conducted. Data requirements for the collection, processing and analysis of the data from the field testing are also specified. Finally, limitations of the research are presented before the chapter is concluded.

The methodology defined by Chapter Three is then used to conduct field testing, the results of which are presented in Chapter Four. Chapter Four begins by discussing changes to the research methodology. The first three phases of the research methodology are then reported on. The first three phases include components of the methodology such as function mapping, test cases and reference sets. The fourth phase of the research, being the analysis phase, is then discussed. The analysis phase presents an analysis of the data collected by the first three phases of the research. Finally Chapter Four presents a summary of the field findings which includes various visual representations of the results.

Chapter Five provides an in-depth discussion of the field findings from Chapter Four. Chapter Five begins by answering the research question and sub-questions that were derived in Chapter Three. The hypotheses for the research question are then tested against the field findings. Chapter Five then presents a discussion of several areas of the field findings. The discussion is split into four areas; testing methodology, evaluating anti-forensic risk, anti-forensic implications and counter-measures and evidential implications. The purpose of the discussion is to discuss the implications of the field findings as well as their significance in the context of digital forensics. Finally Chapter Five recommends possible areas for further research based on this discussion before concluding the chapter.

Chapter Six provides a final conclusion to the research. Firstly, a summary of the research findings are presented, followed by the answers to the research question and sub-questions. Limitations of the research are then discussed and a summary of the recommendations for further research is presented.

Three Appendices at the end of the thesis provide supplementary information. The appendices include test cases, reference sets and the source code for the file fuzzer tool developed during the research.

Chapter Two

LITERATURE REVIEW

2.0 INTRODUCTION

The increased use of electronic media in recent decades has led to the creation of an increasing amount of digital information being created, transferred and stored. Electronic information now pervades every aspect of our society and consequently the digital world also has high criminal usage. Electronic crimes have been steadily increasing in number for years and may continue to do so. Also increasing is the number of regular crimes that now contain an electronic component. One positive side effect of the increasingly electronic nature of crimes is that there is also an increased amount of digital evidence that can be used to prosecute criminals (Casey, 2011, p. 5).

Digital evidence is playing an increasingly important role in the courtroom and to accommodate this increase the digital forensics community have developed standardised methods to collect varied and changing forms of evidence. The development of standardised methods was essential to ensure the reliability, completeness, accuracy, and verifiability of digital evidence (Brown & Kenneally, 2005).

However, the improvement of methods to collect and analyse digital evidence has also resulted in increasingly sophisticated techniques being used to commit electronic crime. A natural outcome of the evolution of electronic crime is the use of techniques to prevent or subvert the creation, storage and analysis of digital evidence. In the forensics community, techniques designed to interfere with creation, storage and analysis of evidence are known as anti-forensics. This thesis is to focus on the rise of digital anti-forensic techniques and Chapter Two presents a review of relevant literature.

The literature reviewed in this chapter will provide context and background for the research presented in this thesis. Literature on a range of areas relating to anti-forensics will be reviewed. Firstly the history of digital forensic tools is presented in Section 2.1. Section 2.1 helps to identify the developments that have led to the creation of forensics and anti-forensics. Section 2.2 reviews

the current research that has been done regarding the anti-forensic risk posed to digital forensics, providing an overview and explanation of the types of anti-forensic risk. In Section 2.3 a case study of past and present anti-forensic risk to Guidance Software's EnCase Forensic is presented. Section 2.3 serves as a practical example of anti-forensic risk detailed in Section 2.2. Literature concerning the legal implications of anti-forensic risk is reviewed in Section 2.4. Section 2.4 translates the technical implications revealed in Sections 2.2 and 2.3 into implications for digital evidence in the court room. Section 2.5 provides a review of methods to evaluate anti-forensic risk to digital forensic tools. Section 2.5 examines possible techniques to mitigate anti-forensic risk and counteract the implications discussed in Sections 2.2 and 2.4. Finally Section 2.6 presents a summary of problems and issues raised by the literature review and Section 2.7 provides a conclusion.

2.1 DIGITAL FORENSIC TOOLS

Section 2.1 provides an overview of how the digital forensics community has arrived at the current situation of extensively using and relying on digital forensic tools. A brief introduction to digital forensics and digital forensic tools is provided first. The introduction is followed by a discussion of the acceptance of and reliance on digital forensic tools.

2.1.1 The Origins Of Digital Forensics

During the late 1970s and early 1980s computer forensics as a field was in its infancy. Charters (2009) describes an "ad-hoc phase" in digital forensics; the field lacked clear processes, procedures, tools and an organised community. The first digital forensic analysts were often law enforcement personal that had some experience using computers. There were no established procedures for how to acquire digital evidence.

The digital forensic community as it exists today originated with law enforcement agencies in the United States during the late 1980's who began working together and forming dedicated units to combat computer-related crimes (Casey, 2011, p. 10). The digital forensic community created standardised methods for undertaking a digital forensic investigation. The standardised

methods were used by dedicated digital forensic investigators who had been trained in undertaking an investigation. Importantly, the digital forensic community also began the development and testing of standardised digital forensic tools.

2.1.2 The Development Of Digital Forensic Tools

In the early era of digital forensics, the amount and type of digital evidence that were examined were small in comparison with the scale and variety seen in investigations today. There were no dedicated forensic tools and often investigations were performed as live analysis using the suspect device to view evidence. The lack of dedicated forensic tools was less than desirable because investigators could potentially be altering evidence inadvertently. The nature of digital evidence leaves it open to being altered or destroyed accidently during collection (Casey, 2011, p. 26). When using the right tools, it is possible to make an exact copy of digital evidence and also determine if digital evidence has been altered (Casey, 2011, p. 26).

During the early 1990s a number of dedicated tools were created such as SafeBack and IMDUMP that were able to forensically acquire a bit for bit copy (a "forensic image") of hard drives without altering any evidence. More advanced tools like EnCase and FTK soon appeared on the market that could not only image a hard drive but also perform complex analysis tasks on forensic images.

Tools like EnCase and FTK represented a turning point in how digital forensic investigations were conducted. Prior to the development of advanced tools, analysis was a very tedious manual process. For example, in order to recover files an investigator was required to have a good understanding of the underlying file system and then manually locate and recover the file using a hex editor. Manual processes were adequate for examining small amounts of data such as a few files on a floppy disk, but as soon as investigators began encountering large volumes of data, the need for task automation became obvious. The strength of tools like EnCase and FTK is their ability to automate large, complicated tasks. Using advanced tools, it is easy to perform tasks such as recovering all deleted files from a hard drive.

2.1.3 Acceptability Of Digital Forensic Tools

Digital forensic tools like EnCase and FTK are universally accepted by legal systems around the world. A common line of questioning of an expert witness starts by asking what tool they used and if the tool is accepted as a suitable or high quality tool by the forensic industry. The acceptability of a common tool like EnCase is rarely questioned any further in the court room. The universal acceptance of digital forensic tools in the court room likely exists due to a lack of technical understanding by the legal community and the general public. An expert witness can usually assume their use of EnCase of FTK will not be challenged to any significant degree. However, some in the forensic community such as Sommer (2010) are concerned that many digital forensic tools have not been tested and proven in the same way as has been done in other forensic sciences.

2.1.4 Reliance On Digital Forensic Tools

Prior to the development of digital forensic tools an investigator would manually analyse all of the evidence at file system level. The increasing size and complexity of digital evidence means that a time-consuming manual analysis is not the preferred approach. Digital forensic practitioners commonly rely on tools to perform complex forensic analysis processes for them. The complexity of digital evidence coupled with the ease with which tools can perform analysis has resulted in a situation where many practitioners have only a minimal level of knowledge about how a tool gets its results.

The danger of this reliance on tools is that the ability to analyse and produce valid evidence is now dependent on a small number of vendors maintaining their tools. Reliance on tools can place investigators in a dangerous position where they cannot be sure if the tool is reliable or they may know the tool is not reliable but the vendor will not fix the issue. Carrier (2002) has advocated the use of open source forensic tools to help mitigate some of the risk of relying on tools.

An example of the danger of this reliance can be demonstrated by flaws discovered in EnCase's handling of times and dates. Several flaws were discovered that likely resulted in expert witnesses presenting incorrect dates and times as evidence in court (Ayers, 2009). The danger of flaws such as that

discovered by Ayers (2009) is that they affect a large portion of digital evidence presented in court because of the reliance on EnCase.

2.2 ANTI-FORENSIC RISK

This Section describes how the digital forensics community has approached the topic of anti-forensic risk in the past. Firstly types of anti-forensic risk will be discussed, then an analysis of the risk factors that allow anti-forensic techniques to succeed and potential mitigation strategies will be presented.

In the last decade many researchers have attempted to define, understand and combat the problem of anti-forensic risk. The researchers all approached the topic from a different point of view and with different motives and goals. The varying approaches have resulted in differing opinions about what "anti-forensics" actually is. Harris (2006, p. S45) attempted to unify research about anti-forensics into an appropriate definition and considers anti-forensics to be "any attempts to compromise the availability or usefulness of evidence to the forensics process". Rogers (2005, p. 3) offers a similar definition, defining anti-forensics as "attempts to negatively affect the existence, amount and/or quality of evidence from a crime scene, or make the analysis and examination of evidence difficult or impossible to conduct." For the purposes of Chapter Two, the definition provided by Harris' will be used when referring to anti-forensics.

2.2.1 Types Of Anti-Forensic Risk

The differing definitions of "anti-forensics" resulted in different classifications for the various types of anti-forensic risk that exist. Similar to the unified definition, Harris (2006, p. S45) also created a unified list of the types of anti-forensics consisting of four types of anti-forensic risk that will be described below.

2.2.1.1 Destroying evidence

The first type of risk, destroying evidence, encompasses any process which destroys evidence or makes the evidence unusable from an investigative point of view (Harris, 2006, pp. S45-S46). An example of evidence destruction would be wiping a hard drive with zeros or blanking all timestamps on files. Simple approaches to destroying evidence like wiping are a common type of anti-forensic risk encountered by digital forensic practitioners; a suspect realises they might be

investigated and then frantically tries to destroy all evidence. On many storage devices and file systems, simply deleting a file is not sufficient to destroy evidence so overwriting techniques must be used. Garfinkel (2007, p. 2) states there are three basic modes of operation for overwriting; overwrite the entire media, overwrite individual files or attempt to overwrite already deleted files by writing to the free space on the media. Harris (2006, p. S46) notes that the tools or processes used to destroy evidence often create new evidence trails. There have been research studies conducted into the tendency of evidence destruction tools to either leave behind evidence or not fully destroy evidence. An example of trails left behind by anti-forensic tools is a study investigating the effectiveness of erasure tools that showed anti-forensic tools does not sufficiently destroy some evidence (Chiang, Triton, & Woodward, 2010). Another study showed that similar anti-forensic tools not only did not sufficiently destroy evidence but also left unique signatures behind that could be used to identify the tool used (Geiger, 2005).

2.2.1.2 Hiding evidence

Hiding evidence is the process of making evidence less visible to the investigative process. The evidence still exists and is not altered, it is just made harder to find. Examples of hiding evidence include changing a document's file extension from *.docx to *.jpeg, or placing a document into a password protected zip container. Simple techniques for hiding evidence are a common type of anti-forensic risk often seen in investigations and are usually easily thwarted. There are some more advanced techniques for hiding evidence that are well known and understood in the forensic community such as the use of host protected areas (HPA) and drive configuration overlays (DCO) on hard drives. Data stored in the HPA or DCO is not visible to the basic input/output system (BIOS) or operating system but can be extracted using specialised tools (Garfinkel, 2007, p. 4). Due to the numerous possible locations and methods of hiding digital evidence more advanced types of evidence hiding can be very effective. However, even if the hidden evidence cannot be found, the possibility that evidence may have been hidden coupled with the presence of hiding tools has been presented as evidence itself (McCullagh , 2005).

Harris (2006, p. S45) states that some attacks on forensic tools and processes could also be considered an evidence trail obfuscation technique. The ability to attack tools to hide evidence trails results in some overlap between anti-forensic techniques in the counterfeiting evidence (Section 2.2.1.4) and hiding evidence categories. An example of an overlap could be an attack against EnCase defect #38041. Defect #38041 prevents keywords from being found in a search when they break across a line due to the text style being used (Bunting, 2010).

2.2.1.3 Eliminating evidence sources

The third type of risk; eliminating evidence sources, is the process of preventing the creation of evidence. Harris (2006, p. S46) notes that if no evidence is created then there is no need to try to destroy or hide the evidence. An example might be an employee who is trying to smuggle sensitive documents out of his company. The employee could connect a USB drive and copy the documents to it; however, connecting the device might leave evidence in locations like the Windows registry that records that a device was connected. Instead, the employee might opt to photograph the documents on his screen with their smartphone thus preventing evidence from being created. Another example of eliminating evidence sources is the use of live CDs or virtual machines; these tools allow anyone to use a computer while containing and minimising the amount of forensically useful information that is left behind (Garfinkel, 2007, p. 5). Harris (2006, p. S46) mentions that in cases where there are little or no evidence trails, the lack of an evidence source could be important evidence in and of itself. An example is a hacker who exploits a system and avoids using techniques that could be noticed and recorded by an intrusion detection system. The fact that the hacker knew what techniques to avoid using may suggest a level of skill which could help narrow down the potential list of suspects.

2.2.1.4 Counterfeiting evidence

The final type of risk is “counterfeiting evidence” which is described as creating fake evidence that appears to be something else; often with the intention of invalidating actual evidence or misleading the investigator (Harris, 2006, p. S46). Importantly Harris (2006, p. S46) includes evidence designed to attack weaknesses in tools and processes under the counterfeiting evidence category.

The classification of issues with tools as counterfeiting evidence is a deviation from earlier work where attacks against tools and processes are in a separate category of their own (Rogers, 2005). A simple example of counterfeiting evidence is the editing of timestamps to make it appear as if a document was created earlier than it was. An example of an attack against a tool is deliberately creating a document with a character set that is not searchable by a particular tool. An expert witness may be unable to explain why the document was not found by a keyword search. Attacks on tools create doubt about the tool and the processes used by the expert witness and may discredit actual evidence that has been found. An advanced example of counterfeiting evidence is manipulating a JPEG image and using sophisticated techniques to hide the history of the modified image (Stamm, Tjoa, Lin, & Liu, 2010).

Harris's (2006, p. S46) definition and classification of counterfeiting evidence is acceptable for the most part. However, the use of the term "counterfeit" implies that false evidence has been created and is being represented as something it is not. Evidence does not necessarily have to be faked, created or misrepresented in order to pose a threat to tools and processes. The category could possibly be renamed "misleading evidence" meaning any evidence that could adversely affect the investigation.

2.2.2 Anti-Forensic Risk Factors

Harris (2006, p. S46) suggests that anti-forensic techniques rely on weaknesses within digital forensics in order to succeed. Harris identified three key problem areas that are exploited by anti-forensics; human factors, dependency on tools and processes and physical and logical limitations of the investigation.

2.2.2.1 Human risk

The first anti-forensic risk factor, the human factor, refers to how an investigator behaves when encountering an anti-forensic technique being used against them. Harris (2006, pp. S46-S47) includes aspects such as the alertness of the investigator and the investigator's training and experience as aspects of the human factor. The "alertness" aspect of the investigator could be expanded to encompass the current mental state or well-being of the investigator. It is not uncommon for an investigator to work all night imaging a computer only to get the evidence back

to the lab and be instructed that analysis needs to begin immediately. Less than perfect working conditions for the investigator lend themselves to a situation where the investigator is not operating at 100% of their mental ability; they are tired and stressed and could easily miss something obvious that a fresh mind would not miss. Rogers (2005, p. 12) notes that attacks on processes and tools (discussed in more detail in Sections 2.2.2.2 and 2.2.2.3) can be subtle and the more automated an examination is the more likely it is to be attacked; an investigator who is not very alert could miss subtle anti-forensic risks. The training aspect is also important as training can help an investigator identify an anti-forensic attack as well as provide some idea of how to mitigate the risk to the forensic process. For example, commercial anti-forensic tools often leave behind tell-tale signs (Geiger, 2005, p. 9); if an investigator is not familiar with which signs to look for they could overlook the use of an anti-forensic tool.

Harris (2006, p. S47) stresses the importance of the experience factor, stating that an investigator relies on his experience to inform his intuition. Investigators continually rely on their past experience to modify the way they investigate. For example, if an investigator had previously encountered a case of a suspect deliberately modifying MRU (Most Recently Used) lists in the Windows registry, they would be more likely to look for the technique being used in future cases.

Harris (2006, p. S46) also notes that an investigator who is naturally inquisitive is more likely to follow unusual or interesting challenges in an investigation. The importance of an inquisitive nature is that the more inquisitive an investigator is, the less likely they are to miss an anti-forensic technique being used.

The discussion by Harris (2006, pp. S46-S47) regarding the way an investigator approaches an investigation demonstrates ways in which the evidence can guide the behaviour of an investigator. Someone who is about to be investigated may be able to deliberately create evidence that guides an investigator away from the evidence of value. The manipulating of people to take some specific action in their lives is known as social engineering (Hadnagy, 2010, p. 10). A possible social engineering attack could be to create a PDF file that looks like a critical piece of evidence. The PDF file could contain a malicious payload that exploits vulnerabilities in Adobe Reader. Creating a malicious PDF

file containing an exploit is simple and does not require technical knowledge when using tools like the Social Engineering Toolkit (Kennedy, 2012). The investigator comes across the file and gets frustrated by the inability of the built-in viewer of their digital forensic tool to view the PDF file. They then export and open the PDF file and the payload is triggered compromising the investigator's computer. Social engineering techniques would be very effective against digital forensic investigators for a number of reasons. Firstly, as Harris (2006, p. S46) noted, most investigators are naturally inquisitive; investigators are trying to find evidence and will be curious about anything suspicious that catches their eye. Secondly the computers used for investigation are typically segregated from external networks and the internet. Computers in a digital forensic lab are not patched or updated as regularly as internet connected computers; investigation computers could contain vulnerable software and outdated anti-virus definition files. Finally, because investigators have their investigation computers isolated, they tend to see them as somewhat immune to security threats. Typically investigators have anti-virus installed but they do not consider the security of their computers to be under any serious threat. The complacency of investigators could result in investigators readily viewing files from a suspect computer using external viewers on their investigation computers. An investigator might be hesitant to launch an executable file from a suspect computer but would be less hesitant to open a PDF file.

Harris's (2006, p. S47) first strategy for mitigating human risk factors is to allow investigators more time to investigate cases that involve anti-forensics. In an ideal world all investigators would like more time to investigate all of their cases regardless of anti-forensics. However time and cost pressures often dictate how long an investigator has to spend on a case rather than the complexity of the case itself. Whitteker (2008, p. 15) notes that time and money are "two of the biggest limiting factors affecting a forensic investigator". Investigators may need to put pressure on the people responsible for determining the time spent on investigations such as managers, clients, judges and lawyers; there is a need to educate people about anti-forensics and why it may make the investigation take longer. Harris's (2006, p. S47) second strategy for mitigating human risks is to ensure that investigators are undergoing continuous digital forensic education as well as being involved in research. The involvement in training and research

would result in increasing the knowledge and experience of the investigator which should help to mitigate human risk. An example of knowledge helping to mitigate human risk is being aware that non-executable files such as PDFs can exploit vulnerabilities in their associated viewer. Rogers (2005, p. 15) agrees that investigators need better education and training and also suggests that if investigators were more knowledgeable about their tools and the underlying processes they would be more likely to identify an anti-forensic risk (Ibid. p. 13).

2.2.2.2 Tool risk

Digital forensic investigators typically rely on one or two tools to conduct their investigation. The reliance on a small number of tools is partly because of the requirement in the community to have standardised tools that can be tested and confirmed to produce reliable results. The small size of the digital forensic community and the barriers to entry for new software vendors also contributes to the lack of diversity in forensic software. The dependency on a few tools from a handful of vendors is good from the point of view of having a set of proven tools but has also made the community vulnerable (Rogers, 2005, p. 9). Garfinkel (2007) split tool risk types into three categories: failure to validate data, denial of service attacks and fragile heuristics.

The first category specified by Garfinkel (2007, p. 81) refers to the risk of tools not properly validating input data before performing a process with that data. A common example of a technique that exploits software that fails to validate input data is the buffer overflow attack. A buffer overflow occurs when a program is writing data to a buffer in memory but overruns the buffers boundary and overwrites the adjacent memory area. The result of buffer overflow is that the program may exhibit erratic behaviour including crashes and memory access errors. In cases where the data being written to memory is under the control of the user it may be possible for the user to control what code is currently being executed and to execute their own arbitrary code. Many investigators have probably experienced an unexpected crash or erratic behaviour when using a digital forensic tool; the crash is likely due to a software bug in the tool that does not properly validate input data. One of the main reasons for the existence of software bugs in digital forensic tools is complexity. Digital forensic tools must be able to acquire data from multiple types of device and then analyse, search and

display thousands of different data formats. A typical software package like Microsoft Word only needs to handle about 20 file formats, whereas in comparison a forensic tool needs to be able to handle hundreds of file formats.

Denial of service attacks in the context of tools refers to the ability of an attacker to exhaust an available resource like memory and CPU time (Garfinkel, 2007, p. 81). Once the resource has been exhausted then the service provided by the tool is denied; the service in the context of digital forensic tools being the specific forensic analysis task being performed. An example of a denial of service attack against digital forensic tools is one commonly referred to as "42.zip". 42.zip is a small zip file that is 42KB in size; however 42.zip contains multiple levels of recursively nested zip files inside itself which, when fully extracted, contain 4.5PB of data (Brinkmann, 2008). Similar malicious compound files were originally used to take down email servers by exploiting the fact that the anti-virus systems on email servers would attempt to extract the zip file and scan its contents (Leyden, 2001). Digital forensic tools also have the ability to extract compound files and, in some cases, extraction is a prerequisite to performing certain analysis tasks. "42.zip" could be used to attack functionality such as the file mounter Enscript, available in EnCase 6. The file mounter Enscript recursively searches through the file system in evidence files and mounts compound files such as zip files. The mounting process consists of extracting the contents of the compound file and storing it within the EnCase case file so that further processes can be run on its contents (i.e. keyword searching). If the mounting process was to encounter 42.zip then the system would keep extracting until it ran out of resources such as hard drive space and memory.

The final type of risk specified by Garfinkel (2007, pp. 81-82) is fragile heuristics which refers to the processes used by digital forensic tools to determine the type or structure of a data object. Essentially, digital forensic tools often have to make educated guesses about what type of data they are processing or how data is structured. For example, when EnCase conducts a file signature analysis it first examines the file extension and file header and then performs a comparison with a list of known signatures (Bunting, 2008, p. 352). The risk of relying on heuristic processes is that they can be easily circumvented by tools such as Transmogrify in order to hide files (Liu, 2008). There is also the risk of a denial of service attack or file hiding technique being possible through the creation of large numbers of

false positives. For example, an attacker could create a large number of text files that start with “PK”. Because text files have no header, a file signature analysis in EnCase will report the text files as zip files (Bunting, 2008, p. 358). The large number of false positives could prevent functionalities such as the file mounter Enscript from working effectively or could divert the investigator’s attention from legitimate zip files containing relevant evidence.

Harris (2006, p. S47) states that tool related risks can be mitigated through two main approaches; firstly using multiple tools and secondly encouraging software vendors to make better tools. The use of multiple tools is a simple solution, however, the cost in time and money of purchasing tools, training and performing the same work twice prohibits many investigators from being able to use multiple tools. The monetary cost factor can be reduced by the use of open source forensic tools which have come a long way in terms of functionality and usability in recent years. Open source tools also have the benefit of any bugs that pose an anti-forensic risk being able to be fixed without vendor involvement. The use of multiple tools also greatly helps mitigate the risk of improperly validated data. However there are only a limited number of ways for digital forensic tools to perform a task which results in tools sharing common methods and techniques; the end result being multiple tools that are vulnerable to the same denial of service and fragile heuristic attacks. There is currently no documentation or suggestion in the community regarding what to do when multiple tools are unable to produce accurate results (Slay & Beckett, 2007, p. 2). A better alternative to mitigating tool risks is to make better and more effective tools. Garfinkel (2007, p. 82) agrees that many anti-forensic techniques can be overcome by improving and fixing bugs in existing digital forensic tools. To get better tools investigators need to place pressure on venders to put more emphasis on fixing bugs in their tools and improving the intelligence behind the processes in their tools. Denial of service attacks such as the use of 42.zip should be intelligently detected and handled by tools. The heuristic systems behind processes like file signature analysis should be improved; for example, tools could look beyond the header and footer of a file and try to identify known file structures within the file in order to identify its type. Investigators could also use various tool evaluation and validation techniques to help identify and mitigate anti-forensic risk associated

with tools. Tool evaluation and validation techniques will be examined in more detail in Section 2.5.

2.2.2.3 Process risk

In the same way as the digital forensics community has standardised tools the community has also developed a standard set of processes for conducting investigations. Standard processes help create consistency and quality across investigations which typically results in better quality evidence being found. It is common for investigators to have a set process for acquiring a hard drive or a set list of processes they perform using a digital forensic tool regardless of the specifics of the particular case at hand. Rogers (2005, p. 9) suggests we may become a "victim of our own success" in standardising forensic processes. Rogers (2005, p. 9) also suggests that there is an assumption that many investigators are "tool monkeys who do not understand what is happening under the hood"; the implication of that being that many investigators blindly follow the same process every time. Many process risks can be viewed as an extension of the human risks discussed in Section 2.2.2.1.

Because the processes an investigator follows are somewhat predictable an attacker can form a good idea of where to target an anti-forensic attack. A simple example of a targeted attack is if an attacker knows that examining INFO2 files is part of an investigator's process, they may take steps to remove or sanitise evidence stored in INFO2 files. A more advanced example would be combining a process risk with a tool risk from Section 2.2.2.2; the attacker could instead create a malformed INFO2 that causes the investigator's forensic tool to crash. Process risks can also be combined with the environmental risks discussed in Section 2.2.2.4; for example, if an attacker knows the investigator's process requires him to perform a full physical acquisition of all hard drives then the attacker could purchase many large hard drives, which would significantly delay the investigation. If investigators' processes are not flexible, or they do not understand the underlying reasons for certain processes, they will be vulnerable to process risks that could greatly increase the cost and time taken to complete an investigation.

2.2.2.4 Environmental risk

Environmental risks are the physical and logical limitations encountered by the investigator caused by investigation environment. Environmental risks are often unexpected and difficult to overcome for an investigator, due to their lack of control over environmental risks.

A physical environmental risk that Harris (2006, p. S48) refers to is the ability of investigators to be able to investigate the latest hardware/software as well being able to investigate antiquated hardware/software. One such example is the micro SATA connector that has become widespread due to the increased popularity of 1.8" hard drives in small form factor notebooks. When the micro SATA connector first appeared many investigators lacked the appropriate adaptors to perform an acquisition of the hard drive directly. The lack of suitable adaptors makes it harder for the investigator to acquire an image and a workaround such as a boot disk is necessary. Initially, a micro SATA connector might not be apparent as an anti-forensic technique. However when referring back to the definition of anti-forensics provided in Section 2.2 it becomes apparent the micro SATA connector does have an impact on the availability of evidence. The impact on availability of evidence can be quantified in the time that the investigator wastes disassembling the laptop, finding out that he does not have the necessary equipment and then having to devise and implement a workaround. Although forensic imaging kits now typically come with micro SATA adaptors the point is that if the investigator does not keep up with the latest technology they are putting themselves at risk.

As Harris (2006, p. S48) suggested, the technology risk also applies in the opposite direction; a suspect might use technology that is so old or exotic that the investigator does not have the equipment or expertise to handle it. For example, it is unlikely that many investigators have a 5¼ inch floppy drive in their imaging kits. The cost and practicality of having equipment to image every type of exotic hardware means an investigator would never have all of the appropriate equipment to handle this scenario. However, the fact that investigators would not typically have appropriate equipment exemplifies the effectiveness of exotic storage media as an anti-forensic attack. A suspect who believes they are going to be investigated could deliberately store evidence across many different types of

storage media. The investigator who tries to image the suspect's digital storage media could discover the following: 8 and 5¼ inch floppies, IBM 3590, SLR and StorageTek T9940 backup tapes and Minidiscs. The typical investigator would struggle to identify the various types of storage even before considering how to perform imaging and analysis. The result of the suspect possessing varied types of storage media is the cost of the investigation in money and time significantly increasing. The final outcome is likely to be that the investigator has less time to perform the analysis and is subsequently less likely to find evidence against the suspect.

One physical environmental factor that is always increasing is the size of storage mediums and the volume of data stored on them. In the forensic community some such as Garfinkel (2010) have predicted that investigators will no longer be able to create a full forensic image due to the increasing size of storage devices. The idea that imaging everything will end is based on the fact that storage devices (magnetic hard drives in particular) have rapidly been increasing in volume while not making much improvement in terms of access speeds. The outcome of the development of large hard drives is that investigators are taking longer to image a typical hard drive. It can be argued that increasingly large hard drives will have a significant effect on standard forensic techniques. There is the possibility of an excessive number of large hard drives being used to deliberately slow down an investigation. For example, a suspect could buy ten large hard drives and fill them with benign content. The suspect could also deliberately buy cheaper "green" models that have slower speeds than typical hard drives. The investigator now has to image ten large hard drives and because they are full of content he cannot gain much benefit from the use of compression. Similar to the previous example, the suspect possessing an unexpected quantity of storage media can greatly impact an investigation in terms of money and time.

Logical environmental factors include issues like knowing how to interpret various types of data structure or being able to process large amounts of data. One example is the difficulty that Carrier (2005, p. 274) had in trying to figure out the NTFS file system. There is no officially published specification on the low level data structures in NTFS, which means no one, other than Microsoft can know if their implementation or understanding of the data structures is correct. The reason the lack of a complete specification is an issue for investigators is that a malicious

user only needs to understand a single aspect of NTFS that may allow an anti-forensic attack; whereas an investigator needs to understand the entire specification in order to know all possible attack vectors (Harris, 2006, p. S48). Another example that Harris (2006, p. S48) gives, is the difficulty that forensic software has in processing large volumes of data. If the investigator manages to image the ten large hard drives referred to in the earlier example he now has to input all of the data into their digital forensic tool and perform an analysis on it. The current analysis approach used by tools is not suited to processing large amounts of evidence (Garfinkel, 2010, p. S68; Richard III & Roussev, 2006, p. 78) which will cause a significant increase in the amount of analysis time required. Similar to the physical limitation, the logical limitation could have an impact on the time and cost of an investigation. The anti-forensic technique of forcing lots of data into a digital forensic tool is only of limited effectiveness; tools are getting better and computer hardware is always getting faster. To make the technique effective the suspect would have to use large volumes of specific data that looks like authentic evidence. For example a suspect could generate 1TB of Internet history which would require a large amount of computational expense to analyse. The full 1TB of Internet history would have to be analysed to ensure any legitimate evidence has not been missed.

Harris (2006, p. S48) notes that investigators will always have to deal with physical and logical limitations and investigators can never make them go away completely. It is likely there will always be an on-going arms race between the ability to create evidence and the ability to process and interpret evidence. Harris (2006, p. S48) suggests that many logical limitations could be mitigated by the use of different approaches to forensic tools such as statistical analysis, mass indexing of data and improving search algorithms. Garfinkel (2010, p. S70) has also encouraged the use of alternative analysis approaches to overcome logical and physical limitations. Richard and Roussev (2006, p. 80) agree that alternative analysis approaches are needed but also suggest increasing the performance of existing analysis techniques. Harris (2006, p. S48) also mentions that there should be improved cooperation from vendors when it comes to understanding how their proprietary technology works.

2.3 REVIEW OF ANTI-FORENSIC RISK IN ENCASE

The EnCase software package created by Guidance Software has become one of the de facto standard tools used in digital forensic investigations. Guidance Software has only one major commercial competitor being Forensic Toolkit (FTK) created by AccessData. A number of smaller competitors in niche areas also exist, however, they do not usually compete directly with EnCase. The limited information regarding sales statistics makes it difficult to determine who the market leader is, however, Guidance Software is typically assumed to hold the majority of the market share. Due to the market dominance, EnCase has been a logical target for researchers looking to demonstrate anti-forensic risks. In Section 2.3 the anti-forensic risks that have been demonstrated to be present in EnCase will be investigated.

While Section 2.3 focuses on EnCase many of the issues discussed could also be applied to other commercial tools such as FTK as well as open source tools like The Sleuth Kit (TSK). As stated in Section 2.2.2.2 there is only a limited number of ways a forensic tool can perform an analysis task; the end result being that many forensic tools use very similar techniques to undertake analysis. The similarity in analysis techniques means that often an attack against one tool will also succeed with another.

2.3.1 Newsham, Palmer, Stamos & Burns - iSEC Partners Inc.

One significant piece of research was done not by forensic researchers but by IT security researchers. In 2007, several employees of iSEC Partners Inc. conducted tests against digital forensic tools in an attempt to locate software bugs. To discover software bugs the iSEC Partners team used two techniques, fuzzing of data formats and manual targeted manipulation of data formats (Newsham, Palmer, Stamos, & Burns, 2007).

2.3.1.1 Fuzzing

Fuzzing is the process of providing intentionally invalid data to an application to attempt to trigger an error or fault condition of some kind (Sutton, Greene, & Amini, 2007, p. 22). The iSEC Partners team used fuzzing to create malformed data structures through methods such as randomly replacing single bytes

(Newsham, et al, 2007, p. 4). Two data structures were targeted for testing; individual files and file system structures. Individual files were targeted in an attempt to locate issues with EnCase's built-in file viewers (Ibid, p. 5). File systems and entire disk images were also targeted in an attempt to locate issues with the techniques used to analyse file systems (Ibid, p. 5). The fuzzing techniques used are discussed in more detail in Section 3.1.5.

2.3.1.2 Targeted manipulation

In contrast to the blind nature of fuzzing, the iSEC Partners team also performed deliberate malformation of data structures (Newsham, et al, 2007, p. 5). Targeted manipulation involves having detailed knowledge about the inner workings of a data structure and then malforming data in certain locations with a specific aim. Examples of manual manipulation performed by the iSEC Partners team include malforming data structures inside JPEG files and creating malformed file systems with unusual data structures such as directory loops (Newsham, et al, 2007, p. 5).. The targeted manipulation techniques used are discussed in more detail in Section 3.1.5.

2.3.1.3 Anti-forensic risk identified

The iSEC Partners team were successful in discovering a number of software bugs that resulted in unusual behaviour from EnCase including evidence acquisition being prevented, crashing while searching or displaying evidence, as well as evidence not being displayed (Newsham, et al, 2007, pp. 9-21). Exploiting software bugs would be classified under the anti-forensic risk category of counterfeiting evidence, as they are considered to be denial of service attacks that block analysis and frustrate the analyst (Ibid, p. 3). Importantly, the iSEC Partners team did not find any software bugs that could result in code execution which would have resulted in a much more severe anti-forensic risk (Ibid, p. 2). The iSEC Partners team found similar software bugs in the open source forensic software The Sleuth Kit (Ibid, pp. 5-9).

2.3.1.4 Response from Guidance Software

The response to the discovered software bugs from Larry Gill of Guidance Software was defensive and challenged the significance of the issues discovered (Gill, 2007). Gill (2007) noted that the integrity of the evidence collection and

authentication process had not been compromised. Gill (2007) pointed out several workarounds for the various issues identified and also mentioned that experienced investigators should know how to work around the issues identified.

Gill (2007) makes many comments about how unlikely it is that someone being investigated would exploit any of the software bugs discovered. As noted by the iSEC Partners team, forensic tools are often used to examine evidence from people suspected of computer crimes as well as computer systems that have been compromised by an attacker (Newsham, et al, 2007, p. 27). The implication being that digital evidence is often under direct control of someone who has the ability and motivation to leverage anti-forensic risks in an attempt to disrupt any investigation.

2.3.2 Metasploit Anti-Forensics Project

The Metasploit anti-forensics project is commonly cited in research into anti-forensic risk. The project was most active around 2005-2006 with its main contributor, Vinnie Lau, producing various anti-forensic research and tools. The project now seems to be inactive with the anti-forensic tools merged into the main Metasploit package and the home page no longer existent. The Metasploit anti-forensics project's home page as at 2008 can still be viewed in the Wayback Machine (Liu, 2008). The Metasploit anti-forensic project created three anti-forensic tools known as TimeStomp, Slacker and Transmogrify; all of which were claimed as firsts in the world of anti-forensic tools (Liu, 2008).

2.3.2.1 Timestomp

Timestomp is a tool that allows for the easy modification of all four file timestamps on NTFS file systems (Liu, 2008). The four timestamps are commonly known as MACE values; modified, accessed, created, and entry modified timestamps respectively. Timestamps are important in an investigation for several reasons. Firstly, timestamps help identify when an event happened and, secondly, multiple events can be put on a timeline to help profile someone's activities (Foster & Liu, 2005, p. 4). Finally, if an investigator identifies a file of interest they are likely to look for files with a similar timestamp (Ibid, 2005).

Importantly, Timestomp only modifies what is known as the standard information attribute (SIA) for each file's master file table (MFT) record, but

timestamp information is also available in the filename (FN) attribute (Foster & Liu, 2005, p. 7).

The effect of Timestomp on tools like EnCase is that an investigator can no longer trust any of the timestamps in the file system to be correct. The effect of the inability to rely on timestamps is that any sort of temporal analysis in an investigation is now more difficult. To counter the lack of file timestamps an investigator can look for other temporal information such as access logs (Piper, Davis, & Shenoi, 2006, p. 81).

2.3.2.2 Slacker

In many file systems, including NTFS, all files are allocated blocks of storage of a certain size regardless of the file's actual size. For example, a 400 byte file might get stored in a 512 byte block leaving 112 bytes of storage unused. The left over space is what is commonly known as slack space. Slacker provides the capability to insert files inside slack space on NTFS file systems (Liu, 2008).

The result of the use of Slacker is that someone can easily hide evidence within the slack space of other files. Evidence in slack space is much harder to identify using tools like EnCase. Although many forensic tools have the capability to search slack space, the difficulty is in knowing which files have had their slack space used for hiding other files (Piper, Davis, & Shenoi, 2006, p. 86).

2.3.2.3 Transmogrify

Transmogrify is a tool that can defeat file signature analysis method used by EnCase (Liu, 2008). Forensic tools like EnCase need to be able to identify file types in order to efficiently process evidence; for example, an investigator may want to run a keyword search over PDF files only. As mentioned in Section 2.2.2.2 EnCase conducts a file signature analysis by examining the file extension and file header and comparing the file's signature to a list of known signatures (Bunting, 2008, p. 352). The file header consists of the first few bytes of the file and is commonly referred to as the magic number. Liu shows the example of starting a text file with "MZ" which EnCase file signature analysis will identify as an executable file (Foster & Liu, 2005, pp. 12-13).

The consequence of Transmogrify is that investigators can no longer rely on the file signature analysis performed by EnCase as completely accurate.

Investigators may have to spend longer on an investigation or manually analyse files in an attempt to determine what type they are.

2.3.2.4 Response from Guidance Software

Guidance Software had a more receptive response to the Metasploit anti-forensics project than to the research done by the iSEC Partners team. Liu was invited to the CEIC conference hosted by Guidance Software in 2006 to talk about anti-forensics and the tools he had developed (Liu & Stach, 2006).

Brian Karney, the Director of Product Strategies at Guidance Software, has been quoted as saying "We think it's a good thing. Computer forensics is an evolving field and there will always be people finding new ways to complicate processes. We'll always have communities doing research to bypass traditional methods" (Hilley, 2007, p. 14). Karney also made mention of Timestomp simply being an extension of techniques that have been used for years such as adjusting the time on a computers clock (Ibid, p. 14). Karney notes that Guidance Software are attempting to build counter-measures to anti-forensic tools such as looking at the FN attribute to defeat tools like Timestomp (Ibid, p. 14). Also of note is that Karney mentions that anti-forensic tools are discussed during training on EnCase (Ibid, p. 14).

The Metasploit anti-forensic research may have been better received by Guidance Software because it targeted general weaknesses in common forensic analysis techniques as opposed to being solely targeted at EnCase. Guidance Software is not at fault if EnCase cannot somehow show the original timestamps after the use of Timestomp. In contrast, the research done by the iSEC Partners team pointed to specific issues that exist because of software bugs within EnCase.

2.3.3 Neckar & Ose - Neohapsis Labs

At the Chicago hacker conference THOTCON in 2010, Chris Neckar and Greg Ose of Neohapsis Labs gave a presentation on anti-forensic techniques for malware to avoid analysis; at the end of the presentation they demonstrated an arbitrary code execution vulnerability that was present in EnCase and FTK (Neckar & Ose, 2010, pp. 27-34). The vulnerability was found by fuzzing different file formats supported by the shared components used by EnCase and FTK (McCash, 2010).

2.3.3.1 Arbitrary code execution vulnerability

The vulnerability was in an external module known as Oracle Outside In that triggered when a malicious file was viewed inside EnCase or FTK (McCash, 2010). Oracle Outside In is an external component that is used by software to read, interpret and view hundreds of file formats (Oracle, 2012). The vulnerability appears to be similar to several previous vulnerabilities in Oracle Outside In, such as a buffer overflow caused by a malformed Excel file. (Verisign iDefense Security, 2009).

Forensic software like EnCase and FTK has to be able to support countless different file formats and it is understandable that they would rely on external components to provide file format support. The Oracle Outside In components are used in a wide range of software and vulnerabilities in the software affect all software that uses it, not just EnCase and FTK.

2.3.3.2 Response from Guidance Software

McCash (2010a) posted to the Guidance Software support forums soon after the presentation asking if a fix for the vulnerability was being worked on and also if Guidance Software had plans to search for similar vulnerabilities internally. Stockdale (2010), the Product Manager for EnCase, replied stating that he believed the vulnerability was the same as an already reported vulnerability in Oracle Outside In and that EnCase had been using a patched version of Oracle Outside In in EnCase versions 6.15 and higher. Stockdale's assertion was questioned by McCash (2010a) who confirmed with Neckar and Ose that the vulnerability was different and existed in EnCase version 6.16. Stockdale (2010) then replied saying that he will follow up with Oracle regarding the vulnerability. Neckar (2010) replied on the forum to confirm the vulnerable version as specifically 6.16.1.4. He also mentioned that he had already been in contact with support at Oracle.

2.3.3.3 Response from the forensic community

McCash led the community response with his post on the Guidance Software support forums and his blog post to the SANS Computer Forensic and Incident Response Blog. McCash (2010a) noted that while the vulnerability discovered was particularly concerning, what was more concerning was the possibility of

further vulnerabilities in EnCase. McCash (2010a) gives the theoretical example of there being code execution vulnerability in a file system parser which would result in a malicious payload being executed as soon as a forensic image is loaded into EnCase. Stewart (2010) responded and suggested the incident was "more a cause of alarm for examiners" than for the vendors and noted that examiners should be aware of the complexity of the tools they are using and the data they are analysing.

McCash (2010) gave several suggestions as to what the forensic community should do in response to the issue. Firstly he suggested that the forensic community apply pressure on Guidance Software and AccessData to then apply pressure on their shared vendor Oracle. Secondly McCash (2010) proposed that someone needs to start undertaking fuzzing tests against forensic software because of its niche nature. As support for his argument McCash cites a presentation given by Paul Craig of Security-Assessment.com at Kiwicon in 2009. Craig (2009) performed mass fuzzing against another niche software industry, the scientific research community and discovered a large number of vulnerabilities. McCash (2010) also recommends that forensic examination procedures and configurations are adjusted to account for the possibility that the analysis process may result in arbitrary code execution compromising the analysing computer.

McCash (2010) offers a list of suggestions to provide mitigation against anti-forensic risks including vulnerabilities in forensic tools. Amongst his suggestions are running a host intrusion protection system (IPS) and using operating system protections such as data execution prevention (DEP) and address space layout randomisation (ASLR). McCash (2010) suggests regularly patching forensic analysis workstations and monitoring logs on workstations for anomalies. Stewart (2010) suggests that the technical fixes proposed "may be riskier than leaving a particular vulnerability open". He suggested that EnCase users can prevent vulnerability by opting to not install Oracle Outside In at the expense of a lack of functionality. McCash (2010) offers two suggestions which are typically part of forensic procedures; firstly workstations should only be used to work on one case at a time and be reimaged afterwards, secondly all results should be verified on isolated and different forensic platforms.

2.3.4 Miscellaneous Research

Most of the significant research demonstrating anti-forensic risks associated with EnCase have been covered in the previous Sections. However, there is some minor and less-known research that has been conducted in the area of anti-forensic risk that will be examined in the following Sections.

2.3.4.1 Compression bombs

A compression bomb is a file which expands massively when its contents are uncompressed or extracted. In Section 2.2.2.2 the example of 42.zip was given; a file that is 42KB in size but once extracted, expands to 4.5PB (Brinkmann, 2008). Compression bombs are an ancient technique in the computing world for maliciously crashing applications and systems. Because of the age of the technique and ease with which compression bombs can be created it could be expected that there would be many cases of people either encountering or testing them with EnCase. However there is little literature on how EnCase responds to compression bombs.

Piper, Davis & Shenoi (2006, pp. 81-82) did some testing using EnCase 4.15 with the results showing that EnCase freezes after getting deep enough into a compression bomb. The authors showed that compression bombs are a viable denial of service attack against EnCase. However, compression bombs by themselves and in their traditional format probably do not pose a significant anti-forensic risk to EnCase.

2.3.4.2 The Grugq

The Grugq is a well-known anti-forensic and security researcher who has presented on a wide range of anti-forensic techniques. He is most well-known for his presentation at Black Hat USA 2005 where he explained in detail a number of techniques for modifying file systems to achieve anti-forensic goals (Grugq, 2005). The techniques are similar to those later used by the iSEC Partners team where they showed malformed file systems pose an anti-forensic risk to EnCase (Newsham, et al, 2007, pp. 10-19). The Grugq often makes reference to anti-forensic issues with tools like EnCase or mentions that he is talking to vendors about anti-forensic issues. However The Grugq, has never publically demonstrated anti-forensic risks present in EnCase. The research from The Grugq

is significant because it is relevant to EnCase even if The Grugq has never demonstrated it to be so.

2.3.4.3 Adonis

A simple anti-forensic technique to hide deleted files from EnCase was demonstrated by Adonis (2007). The technique involves removing the string "FILE0" from a file's MFT record. After making the change, EnCase is no longer able to see the deleted file. The technique demonstrated is very simple and easily detectable if the investigator knows what to look for. Adonis (2007) notes that there is not enough innovation in the area of anti-forensics and his simple technique proves that anti-forensic attacks do not need to be complicated. Adonis's technique is innovative because of its simplicity; it is a simple but novel technique that anyone could have devised. The significance of the Adonis technique is in demonstrating that that significant anti-forensic risk can come from simple techniques.

2.4 EVIDENTIAL IMPLICATIONS

The previous Sections have shown there is always some amount of anti-forensic risk involved with digital forensic investigations. Evidence could be altered or destroyed which would result in courts having to make legal decisions based on an incomplete or inaccurate representation of the actual evidence. However, it can be debated how anti-forensic risks and in particular software bugs in forensic software will actually affect the evidential value of digital evidence. There is very little research, opinion or legal precedent regarding how anti-forensic risk will impact on digital evidence. Carrier (2002, p. 1) notes that "To date, there have been few legal challenges to digital evidence, but as the field matures this will likely change". Section 2.4 will examine some of the possible implications and legal challenges of anti-forensic risks on digital evidence.

2.4.1 Digital Evidence

Digital evidence is defined by the National Institute of Justice (2008, p. ix) as "information and data of value to an investigation that is stored on, received, or transmitted by an electronic device". Digital evidence can be considered to be anything digital that might be of use in an investigation. The fact that the evidence

is digital does not change its nature significantly as compared to normal, physical evidence.

Similar to any other kind of evidence, digital evidence has a number of characteristics and requirements that must be met for its acceptance in the court room. In US courts, the Daubert Standard and Rule 702 of the Federal Rules of Evidence are two significant guides in deciding whether evidence produced by an expert witness is acceptable or not. The Daubert Standard has a list of five recommended guidelines for determining the acceptability of evidence from an expert witness:

- Testability – Can the theory or technique used by tested? Can the theory or technique be refuted or falsified?
- Peer review – Has the theory or technique been subject to peer review or publication?
- Error rate – What is the known or potential error rate for the technique used?
- Standards and controls – Do any standards or controls exist relating to the technique used? How well are the standards or controls maintained?
- Accepted by scientific community – Does the relevant scientific or industry community generally accept the technique used?

Rule 702 of the Federal Rules of Evidence helped to transform the Daubert Guidelines into law and it specifies three requirements for the admissibility of evidence from an expert witness:

- The testimony is based upon sufficient facts or data,
- The testimony is the product of reliable principles and methods
- The witness has applied the principles and methods reliably to the facts of the case.

For anti-forensic risk to invalidate or lessen the reputability of evidence it would be necessary to demonstrate how the risks apply to the above requirements.

2.4.2 Authenticity

The iSEC Partners team enlisted the help of law expert Chris Ridder to help understand the evidentiary implications of the issues they had discovered. Ridder (2007, p. 4) states that evidence may only be used in a court if it is authentic and,

once it is found to be authentic, the relevance of the evidence can be considered in the context of the case. In order for evidence to be considered authentic the evidence presented (e.g. the forensic image) must be shown to be the same as that original exhibit that was collected (e.g. the suspect's hard drive). Authenticity is typically proven by the use of extensive chain of custody documentation that shows how unlikely it is that the evidence has been tampered with (Ridder, 2007, pp. 4-5). Ridder (2007, p. 5) states that currently the prevailing view is that evidence obtained from forensic images satisfies the authenticity requirement (i.e. it is the same as the original exhibit).

However, Ridder (2007, p. 9) notes that a forensic image could be found to be inauthentic due to vulnerabilities and the possibility that the forensic images have been altered. Ridder (2007, p. 9) concludes that if forensic tools can be shown to produce inaccurate results they will be vulnerable to being challenged in a court room. Notably, Ridder (2007, p. 8) also mentions that the authenticity of core concepts in digital forensics such evidence hash values could be challenged if risks like code execution vulnerabilities can be shown to be present in forensic software.

Ridder's (2007, p. 9) final conclusion is that there is a low risk of digital evidence being excluded from a court room because of anti-forensic risk associated with digital forensic tools. In order to provide a significant challenge to their authenticity there would need to be a significant number of serious issues identified in forensic tools.

2.4.3 Reliability Of Digital Forensic Tools

Anti-forensic risk may have wide reaching implications in how courts evaluate the reliability of forensic tools. Current forensic tool testing is extensive but focuses on proving that the tool actually performs the processes it claims to and produces valid and accurate results (Ridder, 2007, p. 8). Current testing does not look for software bugs that could result in security vulnerabilities (Ridder, 2007, p. 8). Ridder (2007, pp. 8-9) notes the lack of security testing will probably not result in evidence being excluded; however testing would provide greater assurance to the legal community about the reliability of forensic software.

In the case of the iSEC Partners research, Guidance Software claimed that an experienced examiner would know how to detect and work around the anti-forensic risks identified (Gill, 2007). In order to verify whether an examiner is experienced, courts will often rely on the examiner following industry standard practices or having industry certifications. Because vulnerabilities in forensic software may not be detectable by the examiner Ridder (2007, p. 8) suggests that certification standards for examiners are not a suitable defence against anti-forensic risk. The alternative proposed is that courts must look at the security standards for the forensic tools themselves; strict industry wide security standards for forensic tools would help the tools being seen as reliable (Ridder, 2007, p. 8).

Ayers (2009) discovered several flaws in EnCases handling of times and dates that likely resulted in expert witnesses presenting incorrect dates and times as evidence in court. Ayers (2009) also noted that a process such as altering a timestamp from UTC to a local time zone should be considered "computations" or "assumptions"; the implication being that an investigator cannot present the altered timestamp as being original evidence. Explicitly identifying which evidence is the result of a computation or assumption allows these computations and assumptions to be challenged and tested. Ayers (2009) wondered whether software vendors should be forced to alert the forensic community to the flaws that affect the reliability of forensic tools.

A reasonable challenge could potentially be made against the Daubert standards and controls guideline as it applies to forensic tools. In comparison to other scientific fields, there are few standards or regulations regarding the use of tools in a digital forensic investigation. The closest the forensic community currently has to standards regarding tools is some digital forensic labs having become ISO/IEC 17025 accredited.

2.4.4 Reverse Trojan Horse Defence

The "Trojan horse defence" refers to a suspect accused of performing an action on their computer claiming they are innocent and that a Trojan or some other malicious software performed the action. The defence tactic was successful the first few times it was used, likely due to jurors being unsettled and confused by complex technology they did not understand. The juror confusion in regard to

complex technology can result in jurors finding reasonable doubt when there should be none (Brenner, Carrier, & Henninger, 2005).

A possible legal tactic is to reverse the Trojan horse reasoning and apply it to malicious data (not necessarily single files or executable files) interfering with forensic tools. The investigator could claim that something from the suspect's computer resulted in a forensic tool not behaving as it should. For example, imagine there is a malformed jpeg file from a suspect's machine that causes a crash when viewed with forensic software. The crash could be harmless or it could be a deliberate and targeted anti-forensic attack by the suspect. It would be difficult to determine whether or not it is or is not an anti-forensic attack. A sophisticated enough attack could plant an undetectable rootkit on the investigator's computer without showing signs such as crashing the forensic tool. It would be difficult to prove any sufficiently sophisticated attack had not occurred.

A court would likely dismiss claims that a suspect was using sophisticated anti-forensic techniques without convincing evidence to back it up. Consider a world where sophisticated anti-forensic attacks are the norm and expected. In the same way the defence can suggest there was a Trojan present on the suspect's computer, the prosecution might be suggest the suspect is deliberately using advanced anti-forensic techniques. As with the Trojan defence, the prosecution do not necessarily need to provide conclusive evidence or prove that their claims of anti-forensic techniques being used are correct; they just need to create reasonable doubt. For example, it is common for forensic investigators to note the presence of wiping tools in their reports regardless of whether there is proof that the wiping tool was used. Evidence of the presence of anti-forensic tools can often strengthen a case against a suspect.

2.5 TOOL RISK EVALUATION

As previously mentioned in Section 2.4.2 Ridder (2007, p. 8) stated that the current testing done in the forensic community is inadequate to find security vulnerabilities in forensic software. The same belief is also shared by the iSEC Partners Team who note that current testing focuses on countering data hiding

techniques and ensuring accurate reproduction of the evidence (Newsham, Palmer, Stamos, & Burns, 2007, p. 1).

The focus of testing in the industry can be demonstrated by asking a forensic investigator whether their write blocker has been tested to block write attempts or if their forensic tool can detect whether there is an HPA or DCO present on the drive. The investigator will be able to give an affirmative answer, point to a wide range of testing done by others, and in many cases, they will have also done testing themselves. If the investigator is then then asked if their forensic software is likely to be free of software bugs that could cause security vulnerabilities they will not be able to give a conclusive answer. Some investigators might readily give an affirmative answer as some in the forensic community do not like to consider forensic software containing security-related software bugs. However, due to the lack of testing, it is difficult to conclude whether forensic software is secure or insecure.

To counter anti-forensic risk (particularly software bugs resulting in vulnerabilities) the forensic community needs an adequate way to test for and evaluate anti-forensic risk. Section 2.5 will examine some of the existing testing and evaluation methods available and how appropriate they are for evaluating anti-forensic risk.

2.5.1 Computer Forensics Tool Testing Program

The most well-known testing in the forensic community has been undertaken by the Computer Forensics Tool Testing Program (CFTT) at NIST. The CFTT has created a specific methodology for testing forensic tools known as the “General Test Methodology for Computer Forensic Tools”.

2.5.1.1 Methodology

The CFTT methodology has the goal of providing assurance to law enforcement personnel that forensic tools can be used for the purposes of a computer forensic investigation (National Institute of Standards and Technology, 2001, p. 1). The CFTT methodology consists of first establishing categories of forensic requirements and then populating the categories with specific technical or functional requirements (National Institute of Standards and Technology, 2001, pp. 1-2). Once requirements are written, test assertions can be written that

describe what a tool needs to be able to do to meet a requirement; the requirement is then translated into a specific test case for a tool to test the assertion (National Institute of Standards and Technology, 2001, p. 2). There are detailed guidelines around the method used for testing that are based primarily on ISO/IEC 17025 and ISO/IEC 13210 (National Institute of Standards and Technology, 2001, pp. 2-3). However there are not any significant limits placed on exactly how an assertion is tested as long as the method can be shown to be suitable for its intended use (National Institute of Standards and Technology, 2001, p. 3). A more detailed review of the CFTT methodology is provided in Section 3.1.1.

2.5.1.2 Usefulness for evaluating anti-forensic risk

The General Test Methodology for Computer Forensic Tools could be adapted to evaluate the likelihood of anti-forensic risk being present in tools. The adaptation would involve defining requirements such as a specific component of a tool being free of severe security vulnerabilities. However, there may be some difficulty in agreeing on how to determine if a tool has meet the requirement and which method is to be used for testing. There is a requirement that the outcome of the testing be measurable in some useful way; it is difficult to quantify something such as how likely a tool is to contain software bugs. The methodology used by the CFTT is better suited to more quantifiable testing, where it is easier to determine if the requirements have been met. However, it would not be infeasible to create new anti-forensic risk requirements in the General Test Methodology for Computer Forensic Tools.

2.5.2 Scientific Working Group On Digital Evidence

The Scientific Working Group on Digital Evidence (SWGDE) has created a similar document to the CFTT known as the Recommended Guidelines for Validation Testing. Unlike the CFTT's methodology, which is designed for use by testing professionals, the SWGDE guidelines have been created for use by any organisation that performs digital forensic examinations (Scientific Working Group on Digital Evidence, 2009).

2.5.2.1 Methodology

The SWGDE guidelines give advice on how to evaluate "if a tool, technique or procedure functions correctly and as intended" (Scientific Working Group on Digital Evidence, 2009, p. 2). The SWGDE guidelines define simple test plans that consist of four main elements; purpose and scope, requirements, methodology and expected results. Within the four elements there are not the strict ISO backed requirements of CFTT methodology; everything is simple and easy for anyone to understand. A more detailed review of the SWGDE guidelines is provided in Section 3.1.2.

2.5.2.2 Usefulness for evaluating anti-forensic risk

Similar to the CFTT methodology, the SWDGE guidelines could also be adapted for use in evaluation of anti-forensic risk. While the CFTT methodology is more rigorous, the SWDGE guidelines provide a simplicity and flexibility that would be needed when evaluating anti-forensic risks. The only issue to overcome with the SWDGE guidelines is that it is primarily aimed at testing functionality and the expected results are typically either pass or fail. Giving definitive results when evaluating anti-forensic risk is not always possible and could encourage testing only that which is easily testable rather than that which is significant.

2.5.3 Digital Forensics Tool Testing Images

Carrier (2010) has created a number of sample forensic images to test functionality in digital forensic tools in an attempt to provide tests that could be easily conducted by the public.

2.5.3.1 Methodology

The sample forensic images are designed to test small and simple test cases and are publically available on the Digital Forensics Tool Testing Images website. The images can be downloaded and tested by anyone in their forensic tool of choice. Carrier details the expected results and also encourages testers to report their results.

2.5.3.2 Usefulness for evaluating anti-forensic risk

Carrier's (2010) approach could be adapted to providing test images for evaluating anti-forensic risk in digital forensic tools. For example, a sample image could be created that contains an NFTS partition with a heavily malformed MFT table or an image could be created that contained thousands of malformed JPEG files. Carrier's (2010) approach is limited by the fact that it is difficult to provide images that cover the wide range of test cases necessary to evaluate anti-forensic risk. It is likely infeasible to provide test images on a large enough scale to do significant testing for anti-forensic risk. A better approach may be to make images that, for example, contain a few dozen JPEG files all of which are malformed in a way that is known to have resulted in crashes in other software that parses JPEG files.

2.5.4 Software Security Testing

Software security testing is a well-established and complex field that has testing techniques that may be practical for detecting anti-forensic risk. The anti-forensic risks identified earlier by the iSEC Partners team (Newsham, Palmer, Stamos, & Burns, 2007, p. 1) were found using the technique of fuzzing. The code execution vulnerability discovered by Neckar and Ose was also found using fuzzing (McCash, 2010). Section 2.5.4 will examine security testing techniques and their possible application in testing for anti-forensic risk.

2.5.4.1 Static analysis

Static analysis refers to either the source code or the disassembled binary of an application being examined but not executed (Klein, 2011, p. 4). Static analysis is essentially the process of looking at the code for the software and trying to find bugs. Static analysis approaches vary and can include going through the code line by line, identifying and following the flow of user influenced input or simply looking for the use of typically unsafe code functions (Klein, 2011, pp. 4-5).

Static analysis only requires the ability to understand source code (if available) or the disassembled machine code. Static analysis is easy in that it does not require extensive setup time or familiarity with a wide range of tools. The difficulty of static analysis lies in having the knowledge and ability to understand the source

code and identify bugs. Static analysis is also a very manual and time consuming process.

2.5.4.2 Dynamic analysis

In contrast to static analysis, dynamic analysis involves the execution of an application and subsequent analysis of the running machine code of the application's behaviour. The benefit of dynamic analysis is that the Section of code currently being executed can be examined on its own without concern for other pieces of code that may not get run and may not be relevant to whatever function is currently being executed. Dynamic analysis makes it easy to perform a certain function in an application (e.g. opening a file) and then to analyse the resulting flow of code of execution. Typical dynamic analysis techniques involve executing software in a controlled environment where a debugger can be attached to allow for monitoring and control of code execution.

A dynamic analysis technique that was discussed earlier is fuzzing where invalid data is provided to software in an attempt to trigger an error or fault condition of some kind (Sutton, Greene, & Amini, 2007, p. 22). Fuzzing is similar to traditional boundary value analysis where the boundaries between good input data and bad input data are tested. The advantage of fuzzing over boundary value analysis is that it can detect bad input data that is not at the boundaries of expected good input data (Sutton, Greene, & Amini, 2007, p. 22). Fuzzing is useful when dealing with a complex application where it may be difficult to determine where and how user-provided input is used by the application (Klein, 2011, p. 5). Many of the issues discovered by fuzzing may not be security critical; however they often provide clues on where to start looking for critical faults (Klein, 2011, p. 5).

2.5.4.3 Source code availability

When performing either static or dynamic analysis it is preferable to have the software's source code available. The original source code improves the quality of testing because it makes it easier to review the source code and tailor tests to the specific design and flow of the software (Carrier, 2002, p. 5). Carrier (2002, p. 5) notes that even if a tool is not open source it would be beneficial if vendors published details about the design and implementation of the functionality of their software.

2.5.4.4 Usefulness for evaluating anti-forensic risk

Software security testing is the most useful method available for locating software bugs that could present anti-forensic risk. Software security testing methods have been proven to be successful in locating anti-forensic risk. One shortcoming of software security testing techniques is that they require a certain level of expertise in programming and software security. The level of skill required to undertake software security testing effectively is not likely to be not present in the wider forensic community. In contrast, the function-based testing of the CFTT and SWGDE methodologies is easily understood and able to be replicated by any digital forensic practitioner. In order for software security testing to be useful in evaluating anti-forensic risk the forensic community may need to rely on outside security experts like iSEC Partners to perform testing. A possible alternative may be to try to simplify and automate software security testing to make it accessible to the wider forensic community.

2.6 PROBLEMS AND ISSUES

The literature reviewed has raised a number of key problems and issues. A summary of these problems and issues are presented below.

2.6.1 Complexity Of Digital Forensic Tools

One of the primary reasons that anti-forensic risk exists is the complexity of digital forensic tools. This complexity is linked to investigations becoming larger and more complicated. Investigations regularly contain many different types of device and require analysis specific to hundreds of unique data formats. To accommodate the scale and variety of evidence, tools like EnCase and FTK have also become larger and more complex. Forensic tools must be able to acquire data from many types of device and then analyse, search and display thousands of different data formats. An earlier comparison was made between forensic tools and a typical software package like Microsoft Word, that needs to handle only a limited number of file formats.

The complexity of forensic tools leads to the possibility of software bugs that potentially pose an anti-forensic risk. The potential for anti-forensic risk was demonstrated by Neckar and Ose (2010, pp. 27-34) who demonstrated a code

execution vulnerability in Oracle Outside In that worked on both EnCase and FTK. The iSEC Partners team did not find any code execution vulnerabilities but they demonstrated that the complexity of forensic software has created a wide attack surface where software bugs could be found (Newsham, Palmer, Stamos, & Burns, 2007).

2.6.2 Approach To Tool Testing

The current approach to testing digital forensic tools is not well suited to testing for anti-forensic risk. Some of the functionality based testing like that done by NIST might coincidentally find software bugs leading to anti-forensic risk. However functionality based testing is not specifically looking for anti-forensic risk.

The software security testing done by iSEC Partners is a promising alternative. However software security testing requires a certain level of expertise that is not typically present in the forensic community. A forensic investigator repeatedly acquires forensic images using the tools and therefore it is easy for him or her to understand how to perform testing of the acquisition functionality. In contrast, a typical forensic investigator would find it difficult to attach a debugger to their tool and look search for software bugs.

2.6.3 Lack Of Research And Testing

There is a significant lack of research and testing for anti-forensic risk, especially regarding risk associated with software bugs. The forensics community has published a small number of papers on the issue. However the existing research mostly deals with defining and describing the problem of anti-forensics. There is very little practical research and testing with regards to identifying anti-forensic risk and assessing its severity.

As mentioned in Section 2.6.2 there is not currently any established methodologies specifically tailored to testing tools for anti-forensic risk. There are the few well known examples of researchers in the security industry exposing anti-forensic risk in tools; however there has not been a significant amount of testing. There is also the issue that, while the security industry can test forensic

tools for software bugs, they may not have the knowledge and expertise to assess the severity of the anti-forensic risk associated with the software bugs.

2.6.4 Attitude Towards Anti-Forensic Risk

The issue of anti-forensic risk typically polarises the digital forensic community. On the one hand there is the view that anti-forensics is a critical threat that will cause the end of traditional digital forensic investigations. On the other hand there is the view that anti-forensics is only a theoretical risk and will never pose a significant threat. The first view cannot be proven to be true because of the lack evidence to demonstrate a significant anti-forensic risk. However the same lack of evidence also means the second view is equally unproven.

Currently it is difficult to determine how severe the issue of anti-forensic risk is. It will be necessary to engage the forensic community in the process of testing and assessing anti-forensic risk. In order to gain support and resources from the wider forensic community the focus of anti-forensic research and testing may need to change from simply breaking forensic tools and showing that risks exist. The forensic community is likely more interested in developing testing methodologies and mitigation strategies.

2.7 CONCLUSION

Chapter Two has investigated and reviewed the current state of anti-forensic risk. An overview of the types and causes of anti-forensic risk has been developed. Chapter Two examined a wide range of anti-forensic risks and the resulting implications. A detailed case study of EnCase specific anti-forensic risk was presented. The case study reviewed a number of anti-forensic risks that other researchers have identified in EnCase. The purpose of the case study was to show the feasibility and possible implications of tool related anti-forensic risk. The case study was following by a review of the possible legal implications of anti-forensic risk. Finally a comprehensive review of possible methods for testing and evaluating anti-forensic risk was presented.

The current state of tool-related anti-forensic risk is that a number of threats have been shown to exist. However the current risks discovered have mostly been relativity minor with very few severe risks identified. The current

risks have gone mostly unnoticed and have not caused any significant disruption to forensic investigations. It is difficult to determine if the risks previously identified are indicative of an endemic problem with digital forensic tools. It is not possible to conclude that anti-forensic risks posed to tools are a significant issue. However it is also not possible to conclude that digital forensic tools are largely safe from anti-forensic risk. More testing is required before a proper determination can be made.

The testing methodologies currently used by the forensic community are inadequate for testing and evaluating anti-forensic risk in tools. The community needs to develop and adopt new ways to test tools and then perform tests on a wide scale. The testing would be beneficial regardless of whether anti-forensic risks are discovered or not. If risks are identified then the forensic community can respond accordingly. If risks are not identified then the forensic community now has evidence for the safety of their tools; the lack of risk would also help reassure the legal community who rely on digital forensic investigators to produce valid evidence.

Chapter Three

RESEARCH METHODOLOGY

3.0 INTRODUCTION

Chapter Two critically reviewed literature from a range of areas relating to anti-forensic techniques and risks. The literature review investigated the extent of anti-forensic risk and the weaknesses in current digital forensic tool testing techniques. The most significant anti-forensic risks were shown to be the result of software bugs in digital forensic tools. In Chapter Three a research methodology will be constructed that will be used to investigate the possibility of testing for software bugs that may cause anti-forensic risk in digital forensic tools. Chapter Three will build upon Chapter Two to present a suitable methodology for research.

Firstly, a review of similar studies is presented in Section 3.1. The analysis of similar studies will help to derive best practices for developing the research methodology. Processes and techniques that are shown to work effectively will be adopted by the research methodology. Research questions derived from Section 2.6 are presented in Section 3.2.2 with the associated hypotheses presented in Section 3.2.3. A breakdown of the research phases is described in Section 3.2.4 and is followed by the data map displayed in Section 3.2.5. Data requirements are defined in Section 3.3 and limitations of the research are discussed in Section 3.4. Chapter Two is then concluded in Section 3.5.

3.1 REVIEW OF SIMILAR STUDIES

In this section five similar studies are critically reviewed and analysed. Section 3.1 will examine how other researchers have defined their research and implemented their methodologies. The five studies have been selected on the basis of their relevance to testing forensic tools and the reputation of the sources.

Sections 3.1.1 and 3.1.2 examine the CFTT and SWGDE methodologies in depth. The CFTT and SWGDE methodologies are two well-established and accepted methodologies for testing forensic tools. In Section 3.1.3 Wilsdon and Slay (2006) propose an alternative approach to testing based on black box testing methodologies. Wilsdon and Slay (2006) have attempted to simplify the

established methodologies for wide spread use by the digital forensic community. Guo, Slay and Beckett (2009) build on the work of Wilsdon & Slay (2006) in Section 3.1.4 and propose a function oriented approach to testing forensic tools. Guo, Slay and Beckett (2009) propose a process-mapping technique to identify the functions of tools that require testing. In Section 3.1.5 the fuzzing methodology used by Newsham et al (2007) to software bugs is examined in detail. Fuzzing was successfully used by Newsham et al (2007) to discover a number of software bugs that had associated anti-forensic risk.

3.1.1 CFTT General Test Methodology For Computer Forensic Tools

The CFTT from the NIST program and their methodology for testing computer forensic tools were discussed earlier in Section 2.5.1. In this Section their methodology will be more critically examined for processes and techniques that can be adopted. The CFTT program is a joint program started by NIST that involves other US organisations including the National Institute of Justice (NIJ). The General Test Methodology for Computer Forensic Tools is one of many initiatives started by the CFTT at NIST to support the testing of tools. This methodology has the goal of providing assurance to law enforcement personnel that forensic tools can be used for the purposes of a computer forensics investigation (National Institute of Standards and Technology, 2001, p. 1). The General Test Methodology for Computer Forensic Tools builds upon various international standards including ISO/IEC 17025 which helps to provide assurance regarding the scientific rigor and merit of the NIST methodology.

The NIST approach to testing tools begins with establishing categories of forensic requirement. The forensic requirement categories are functions of forensic tools that have been determined and grouped by expert users. Grouping functions is done to make it easier to identify a smaller set of requirements that are shared by similar functions. The next step in the methodology is to specify the set of requirements that are required to be met by a forensic tool in one of the previously defined categories. Similar to the creation of requirements categories the requirements specifications are first created by expert users; although the final set of agreed requirements is decided based on a consensus from the forensic community. An example of a category of forensic requirement could be string searching. This category would then be populated with requirement specifications

the forensic tool needs in order to be able to search using one or more character sets.

Once the requirements have been specified the NIST methodology then establishes test assertions and test cases for each requirement. A test assertion is defined as “a statement of behaviour, action, or condition that can be tested or measured” by NIST (2001, p. 2). A test assertion translates a requirement into something that can be tested. An assertion for the previous multiple character set searching requirement would be that the tool returns search results from UTF-8 formatted files. This assertion is then applied and tested through one or more test cases. A test case is defined as “what is to be tested or one instance of what is to be tested” by NIST (2001, p. 2). To test the above assertion of regarding a UTF-8 file a test case could be set up consisting of a sample UTF-8 file, with a known keyword present inside the file. The test case would then search the sample file for the known keyword using the forensic tool and determine whether the correct result is returned.

The test cases and assertions specify what exactly is to be tested. The NIST approach next requires definition of the method used to conduct testing of the test cases. The test method is defined by NIST (2001, p. 2) as “a combination of the software used for testing and the procedures for completing the testing”. The test method is the software, environment and procedures that ensure the test results are valid and repeatable. The NIST methodology has adapted the ISO/IEC 17025 required documentation for non-standard test methods into 16 specific documentation requirements for forensic tool testing methods. These 16 requirements include expected requirements like recording the apparatus used, the software version used and environmental conditions. There is also the requirement to adequately document exactly how the test is to be conducted as well as requirements to identify the expected outcomes and present a strategy for data analysis. The ISO/IEC 17025 standard (1999) requires that any non-standard testing method is suitably validated; for this reason the NIST approach includes publically releasing all details of testing to be judged by the forensic community (National Institute of Standards and Technology, 2001, pp. 2-3).

The NIST methodology also includes specific testing result reporting requirements. Again these requirements are based on ISO/IEC 17025 and require that test results are “reported accurately, clearly, unambiguously and objectively”

(ISO/IEC 17025 Standard, 1999). Essentially the goal of the reporting requirements is to ensure that anyone is able to easily interpret, understand and reproduce the test.

3.1.2 SWGDE Recommended Guidelines For Validation Testing

The SWGDE guidelines were briefly discussed earlier in Section 2.5.2, the SWGDE methodology will be discussed here in more detail. The SWGDE (2009, p. 2) guidelines are designed to determine "if a tool, technique or procedure functions correctly and as intended". The targeted audience for the SWGDE (2009, p. 2) guidelines is specified as "All organizations performing digital forensic examinations". The SWGDE guidelines are designed to be a practical guide for the typical forensic analyst who wants to validate their tools without dealing with the complexity of the CFTT requirements.

The first step in the SWGDE guidelines is the creation of a test plan. This test plan begins with details such as scope, requirements and methodology. Little guidance is given about how the requirements and methodology should be defined which allows much greater flexibility than the CFTT approach. The test plan should then specify test scenarios that are assigned to each requirement. These test scenarios contain details of the testing environment, testing procedures and expected results. Again there is little guidance given regarding how these test scenarios should be defined. Finally, the test plan should include details regarding the test data that will be used in the test scenario; if an existing reference set cannot be used then the process to create a new reference set should be documented.

The next step in the SWGDE guidelines is to perform the test scenarios and create a test report containing the results. The SWGDE guidelines suggest that the tester use equipment which is known to be in a good condition. It is also suggested that equipment has a known configuration which matches the typical configuration the tester would use when performing an actual forensic analysis. The SWGDE guidelines have a process for dealing with anomalies which consists of identifying the cause of the anomaly, verifying the cause of the anomaly and finally adjusting the test scenario, if possible, and re-testing.

The test report should document each individual test scenario as well as noting the pass or fail status for each requirement. Any re-tests or anomalies

should be documented including initial tests that may have been invalidated by an anomaly. The SWGDE guidelines suggest the creation of a summary report which includes an overview of the pass or fail status of the tool or procedure being tested and an overview of the test scenarios. This summary report should also contain recommendations and discussion from the tester. Finally the test report should document all differences between actual results and the expected results so a determination on validation can be made.

None of the documentation and testing required by the SWGDE guidelines has a set format and they can be adjusted as necessary. The SWGDE guidelines include several samples of documents such as test plans and test scenario reports; however these are only samples and do not need to be rigorously followed.

3.1.3 Validating Forensic Software Utilising Black Box Testing Techniques

Wilsdon and Slay (2006, p. 3) acknowledge that the CFTT and SWGDE methodologies are "extremely comprehensive and scientifically sound"; however they also note that these existing testing methodologies are failing to satisfy the demands of the industry. Wilsdon and Slay (2006, p. 3) make the assumption that the current lack of tool evaluation is likely due to a lack of skill, experience, time and finance. It can be inferred from Wilsdon and Slay that the CFTT and SWGDE methodologies are not suitable for meeting the testing requirements of most forensic laboratories. The authors propose a streamlined methodology similar to that of the CFTT and SWGDE that "addresses shortcomings of other frameworks and extends their capabilities" (Ibid, p. 9).

Wilsdon and Slay (2006) propose a six step evaluation process for forensic tools. The first step is to acquire the software to be tested and accurately identify it. This identification is done through the process of creating a signature using techniques such as MD5 hashing. The goal of the signature is to prevent any other versions of the software being thought to have been tested. The next step is to identify the functions of the software; Wilsdon and Slay (2006, p. 7) identify functions as "some component of the software which delivers a result when executed". These functions can be identified by reading the software's documentation, using the software and making inquiries of vendors and the digital forensic community. The identified functions must be clearly documented, including any dependencies between functions.

Test cases can then be created and mapped to the functions. All of these test cases are to be based on black box testing techniques as Wilsdon and Slay (2006, p. 7) assume the software being tested is closed-source and proprietary. Reference sets should be created and used as inputs for the test cases. These reference sets must be designed to test specific functions in isolation and all should test the actual operation of the software in a typical environment. These reference sets can be used to test multiple functions and can also be shared among organisations. The sharing of reference sets would allow other organisations to peer review them and determine if they are suitable for testing the function in real world environments. The test cases and reference sets are focussed on testing a specific function but should also test the software and environment as a whole; this is a departure from the isolated and contained approach of the CFTT and SWGDE methodologies.

The next step is to develop a result acceptance spectrum based on the methodology proposed by ISO 14598.1-2000. The result acceptance spectrum is used to assess the results of the test against the expected outcomes. The results for each test would fall into one of four groups specified by ISO 14598.1-2000; exceeds requirements, target range, minimally acceptable and unacceptable. Wilsdon and Slay (2006, pp. 7-8) note that these groups may differ between environments; for example a military environment may have a different definition of unacceptable to a civilian environment. If a function does not fall into the acceptance range, any dependent functions should also be considered to have failed. This is to eliminate situations where a function passes a test in isolation but actually produces invalid results because it is dependent on a function that has failed.

Once the test cases and result acceptance spectrum have been defined, execution of testing can begin. All testing is to be documented according to ISO 17025-2005 and AS 4006-1992 requirements which is similar to the CFTT methodology. The results from the testing are to be assessed against the result acceptance spectrum and all functions that fall into the acceptance range are to be classified as passed. It is possible for a function to be considered as having passed evaluation even if it does not pass every test case. The idea behind the creation of reference sets is that some of the reference sets are designed to test the limitations of the software. For example, if a reference set contained ZIP files that were

modified outside of the official specification, it would be considered unacceptable if the software was not able to expand them; however the function of expanding ZIP files can be considered to have passed if many other test cases fell into the acceptable range.

The final phase is to release the testing results to the forensic community. Public release allows the community to peer review the method used and benefit from the testing results. Wilsdon and Slay (2006, p. 8) note that once software has been tested and evaluated it does not mean that the software cannot be retested in the future. Importantly any future versions of software or patches do not inherit the evaluation of the tested version. Wilsdon and Slay (2006, p. 8) suggest that vendors would be more likely to release updates less frequently but to a higher standard, if the community relied on their evaluation methodology.

The methodology proposed by Wilsdon and Slay (2006) is essentially the CFTT and SWGDE methodologies adapted for use by members of the wider forensic community who do not have the resources of NIST or large forensic laboratories. The idea of a large set of community-shared reference sets could be considered an expansion of the test images created by Brian Carrier that were discussed in Section 2.5.3. A key benefit of the idea of multiple laboratories testing reference sets is the ability for software to be tested in a wide range of real world environments; this is something that is not easily accomplished by organisations such as NIST or the software vendors.

3.1.4 Functionality Oriented Validation And Verification

Guo, Slay and Beckett (2009) have proposed a functionality-oriented approach to validation and verification of the functionality in digital forensic tools derived from earlier work done by Slay and Beckett (2007). This functionality-oriented approach consists of splitting digital forensic analysis into several categories and then further splitting these categories into sub-categories and components through "process mapping" (Guo, Slay, & Beckett, 2009, p. 2). One of the reasons for developing this new methodology is that Guo, Slay and Beckett (2009, p. 4) believe current methodologies like that used by the CFTT are too vague and broad; they do not offer any conclusive way to determine which functions need to be tested.

The functionality-oriented methodology begins by creating a systematic and scientific description of the digital forensic discipline. Fundamental processes in an investigation are specified such as identification of evidence and then specific processes such as keyword searching are mapped under the fundamental process. For each of the mapped functions a set of requirements are specified; an example given for keyword searching is the requirement to find a keyword in deleted files. From these requirements a reference set and test case can be developed to test and evaluate the requirement. This reference set can be shared with the community to allow for peer review. Where this functionality-oriented approach differs from the earlier work done by Wilsdon and Slay (2006) is the mapping of processes in the digital forensic discipline. This mapping allows the entire discipline to be mapped in terms of processes that can have expected results and the reference sets associated with them. If the proposed methodology could be realised, it would allow for the testing of any tool by simply determining the appropriate function and using the available reference sets to test against the expected results.

The functionality-oriented methodology has currently been applied to the searching functionality by Guo, Slay and Beckett (2009) and later to the forensic copying functionality by Guo and Slay (2010). These two examples of applying the methodology demonstrate that it is feasible to implement the methodology. However, if the methodology is to reach its full potential, it will need wide spread acceptance and adoption by the community. The process mapping of the entire discipline alone will be time-consuming and complex.

3.1.5 Breaking Forensics Software: Weaknesses In Critical Evidence Collection

The research done by Newsham, Palmer, Stamos and Burns (2007) of iSEC Partners Inc. was discussed earlier in Section 2.3.1. In this Section their testing methodology will be examined in greater detail. The iSEC Partners Inc. research represents an outsider's approach to testing forensic tools by IT security researchers. The forensic community is typically testing tools to determine if a certain functionality works; the iSEC Partners team argue that software security should also be a concern as well as functionality (Newsham, Palmer, Stamos, & Burns, 2007, p. 1). The iSEC Partners team conducted testing against forensic

tools and discovered a number of security-related software bugs. To conduct testing two techniques were used; fuzzing of data formats and manual targeted manipulation of data formats (Newsham, et al, 2007, p. 5).

The first of these techniques known as fuzzing was briefly covered in Sections 2.3.1.1 and 2.5.4.2. Fuzzing is providing invalid data to software in an attempt to trigger an error or fault condition of some kind (Sutton, Greene, & Amini, 2007, p. 22). The iSEC Partner's team used fuzzing against a number of data formats and input the resulting invalid data into forensic tools. In its simplest form, fuzzing can consist of simply randomly replacing bytes in a data structure; at its most advanced it requires manipulating specific byte locations with knowledge of the properties of a data structure. The approach to fuzzing used by iSEC Partners was rather simple but not entirely blind and random. A set of mutations were used that are designed to exploit typical programming mistakes commonly found in software. An example of one of these mutations is replacing a sequence of NUL bytes with random values of the same length. Fuzzing was performed on a number of file formats such as JPEG images and PDF documents with the goal of detecting problems with the built-in file viewers in forensic tools. Fuzzing was also performed on the file system structure in an attempt to reveal issues with the methods used by forensic tools to interpret file systems.

The second technique used was manual targeted manipulated of data formats; this was briefly discussed in Section 2.3.1.2. Targeted manipulation is the process of modifying specific portions of a data structure guided by a detailed knowledge of the data structure. Targeted manipulation attempts to trigger the same errors and fault conditions that fuzzing does; however it is generally slower, more complex, but with a higher success rate. The iSEC Partners team performed targeted manipulation of file formats such as JPEG files where they modified data structures related to memory management. Many file system structures were also manipulated manually including creating malformed MBR partition tables and unusual file system structures like directory loops and deeply nested directories.

The techniques used by iSEC Partners were successful in uncovering a number of software bugs in forensic tools. It is unlikely that many of these software bugs would have been detected by traditional function based testing approaches to testing forensic tools; this is because function based testing usually tests input data that has not been malformed.

3.2 RESEARCH DESIGN

The research design will describe and define the approach that has been selected for this thesis. A range of literature has been reviewed and their methodologies have been analysed. Appropriate techniques from the literature will be adapted into the research methodology.

The five studies reviewed in Section 3.1 are discussed in Section 3.2.1 and their implications for the research design are described. Research questions derived from the literature are defined in Section 3.2.2 and the associated hypotheses are described in Section 3.2.3. The four phases of the proposed research methodology are described in Section 3.2.4. Finally a data map is presented in Section 3.2.5.

3.2.1 Review Of Similar Studies

Five similar studies have been reviewed in Section 3.1. The first two reviewed; the CFTT and SWGDE methodologies represent the status quo in testing forensic tools. The CFTT and SWGDE methodologies are tried and tested approaches to testing and evaluating forensic tools. These approaches can be summarised as identifying a function the forensic tool should perform and then rigorously validating that function.

However, as pointed out by Wilsdon and Slay (2006, p. 3), the CFTT and SWGDE methodologies have failed to satisfy the testing demands of the forensic community. Wilsdon and Slay (2006) do not propose any radical change to what is tested; testing requirements are identified in much the same way as in the CFTT and SWGDE methodologies. The authors focus on how testing should be conducted and propose a simplified and streamlined methodology for testing that allows for and encourages widespread testing by the community. The approach suggested by Wilsdon and Slay (2006) uses reference sets and community involvement to greatly increase the scale of testing being performed.

Guo, Slay and Beckett (2009, p. 4) later stated that they believe current testing methodologies are too vague and broad. They suggested that current methodologies do not offer any conclusive way to determine what functions need to be tested. Methodologies like the CFTT and SWGDE approaches are targeted at whatever functions are deemed to be important; no attempt is made to figure

out what should be tested. A consequence of the CFTT and SWGDE approaches is a large focus on functions like disk imaging while functions such as built-in document viewers remain untested. Guo, Slay and Beckett (2009) propose a process-mapping methodology that would allow for the mapping of all functions of forensic tools. Process-mapping would assist in revealing functions and areas of forensic tools where little or no testing has been performed.

The current methodologies used by the CFTT and SWDGE along with changes proposed by other researchers are well suited to validating functions of forensic tools. However, Newsham et al (2007, p. 1) argue that more security-focused testing of forensic tools is required. Current methodologies focus on testing enough use cases to ensure that a function can be validated as working. Security testing requires that enough use cases are tested to conclude the function does not pose a security risk regardless of whether the function works or not. Newsham et al (2007) demonstrated the use of fuzzing to successfully locate a number of security-related issues with forensic tools. Fuzzing an easy to understand and perform in comparison to more complicated security testing, such as disassembling executable files and analysing assembly code.

The main implication of the reviewed literature is that the current testing methodologies like the CFTT and SWGDE are not suitable for community-based testing or security-related testing. The simplified approach suggested by Wilsdon and Slay (2006) could greatly increase the amount of testing done by the forensic community. In particular, the use of reference sets would make testing digital forensic tools much more practical. The function mapping proposed by Guo, Slay and Beckett (2009) could assist in identifying tool areas that have not been adequately tested; these areas are prime candidates for software bugs. Finally, the fuzzing methodology used by Newsham et al (2007) represents a novel technique for testing for security-related software bugs. Fuzzing has been widely used for software testing; however it has not been used in a significant way for testing digital forensic tools.

3.2.2 Research Questions

The literature review in Chapter Two and, in particular, the problems and issues identified in Section 2.6 provide a basis for establishing the research question of the thesis. Section 2.3 showed that tools such as EnCase contain some software

bugs that present an anti-forensic risk. Section 2.6.1 argues one reason for these software bugs is the complexity of forensic software; however, Section 2.6.4 notes that not enough testing has been done to conclusively determine the scale and scope of the anti-forensic risk. The research question for the thesis is thus:

What is the anti-forensic risk caused by software bugs in digital forensic tools?

In order to sufficiently answer the research question, a number of sub-questions need to be developed. As noted in Sections 2.6.2 and 2.6.3 a key issue in determining the extent of anti-forensic risk caused by software bugs is the approaches currently used for tool testing as well as the lack of testing being performed. Section 2.6.4 also suggested that more involvement from the forensic community will be needed to make any significant progress in testing for and eliminating anti-forensic risk. Several sub-questions can be derived from the research question as well as the issues discussed in Section 2.6:

Sub-question 1: What testing approaches are appropriate to test for the presence of software bugs in digital forensic tools?

Sub-question 2: What test cases and reference sets can be developed to promote and assist the forensic community in testing for and evaluating software bugs associated with anti-forensic risk?

Sub-question 3: How can software bugs in digital forensic tools be ranked and evaluated in terms of severity and anti-forensic risk?

Sub-question 4: What are the risks caused by the presence of software bugs in digital forensic tools?

3.2.3 Hypotheses

The hypotheses for the anti-forensic risk caused by software bugs in digital forensic tools are as follows:

H1: No software bugs are detected.

H2: Software bugs are detected but they do not present an anti-forensic risk.

H3: Software bugs are detected that present a minor anti-forensic risk.

H4: Software bugs are detected that present a critical anti-forensic risk.

The difference between a minor and critical anti-forensic risk is that a critical anti-forensic risk has the potential to compromise the evidence or the security of the examiners machine. In contrast, a minor anti-forensic risk only prevents or disrupts the analysis of evidence.

3.2.4 Research Phases

The proposed research is loosely adapted from the black box testing techniques developed by Wilsdon and Slay (2006) and consists of four phases (Figure 3.1). The first phase consists of mapping functions using a similar approach to the method developed by Guo, Slay and Beckett (2009). Function mapping will assist in identifying a number of areas in digital forensic tools that have not been extensively tested. The research done by Newsham et al (2007) will also be used to help inform the decision regarding which areas of digital forensic tools are to be tested. Once areas have been targeted for testing, specific requirements for testing can be created. The testing requirements specify which criteria need to be met for the test to be considered a pass. The test requirements focus on two key criteria; functions completing accurately and the handling of malformed input data.

The second phase focuses on setting up and preparing test cases. The test cases are derived from the testing requirements identified in Phase 1. The test requirements specified in Phase 1 will be high level and may not be directly testable; the test cases are designed to translate the requirements into something practical and testable. Associated with the test cases will be a number of reference sets containing malformed data. The reference sets will be created using malformation techniques adopted in Newsham et al (2007). Also associated with the test cases are acceptance spectrums adapted from those proposed by Guo, Slay and Beckett (2009). The acceptance spectrums allow a range of possible results for each test case that extends beyond the simple pass or fail mechanic of traditional forensic tool testing.

The third phase consists of performing the tests described by Phase 2. The tests and results are to be extensively documented. The results are then analysed and interpreted in Phase 4. Finally, the results and reference sets are to be

publically released as proposed by Guo, Slay and Beckett (2009). The public release will allow the forensic community to benefit from the research as well as use the reference sets to both peer review the results and conduct testing in differing environments.

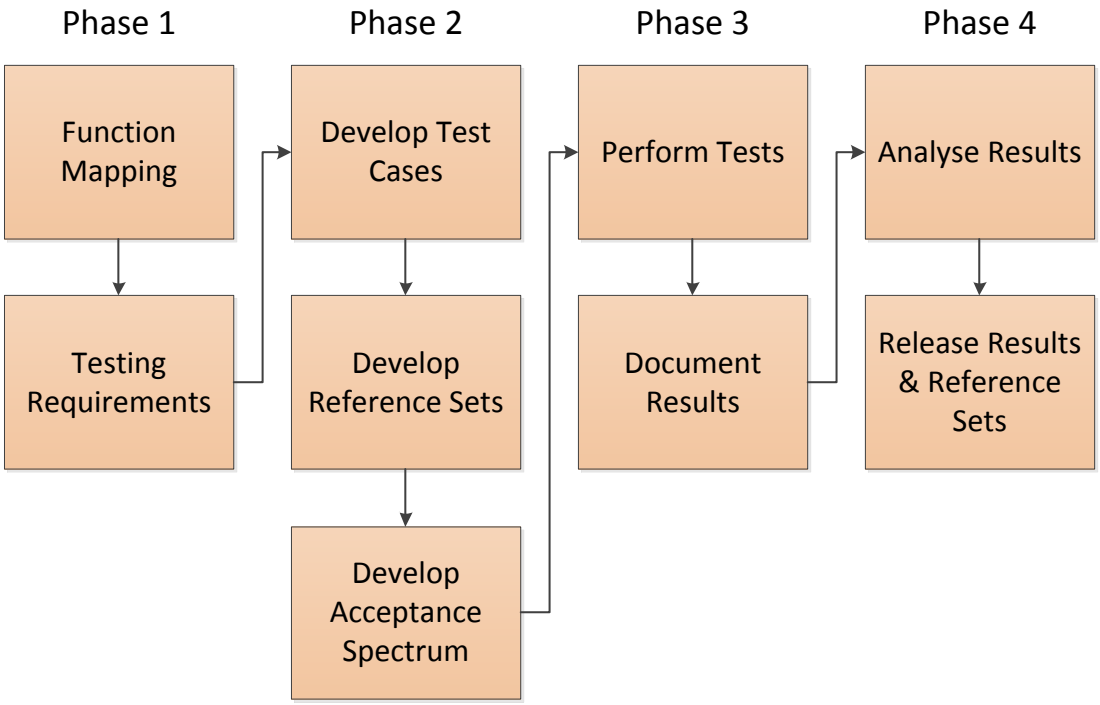

Figure 3.1: Research phases

Figure 3.1 shows visually how the tasks are divided between each research phase. Also shown are the predecessor relationships between tasks; for example, testing requirements must be completed before test cases can be developed.

3.2.5 Data Map

A data map is presented in Figure 3.2 that shows the data relationships between the components of the research methodology. The research question is first related to the research phases described in Section 3.2.4. The research phases are then related to the data collection and analysis requirements that are discussed in Section 3.3. The data collection is then related to the hypotheses.

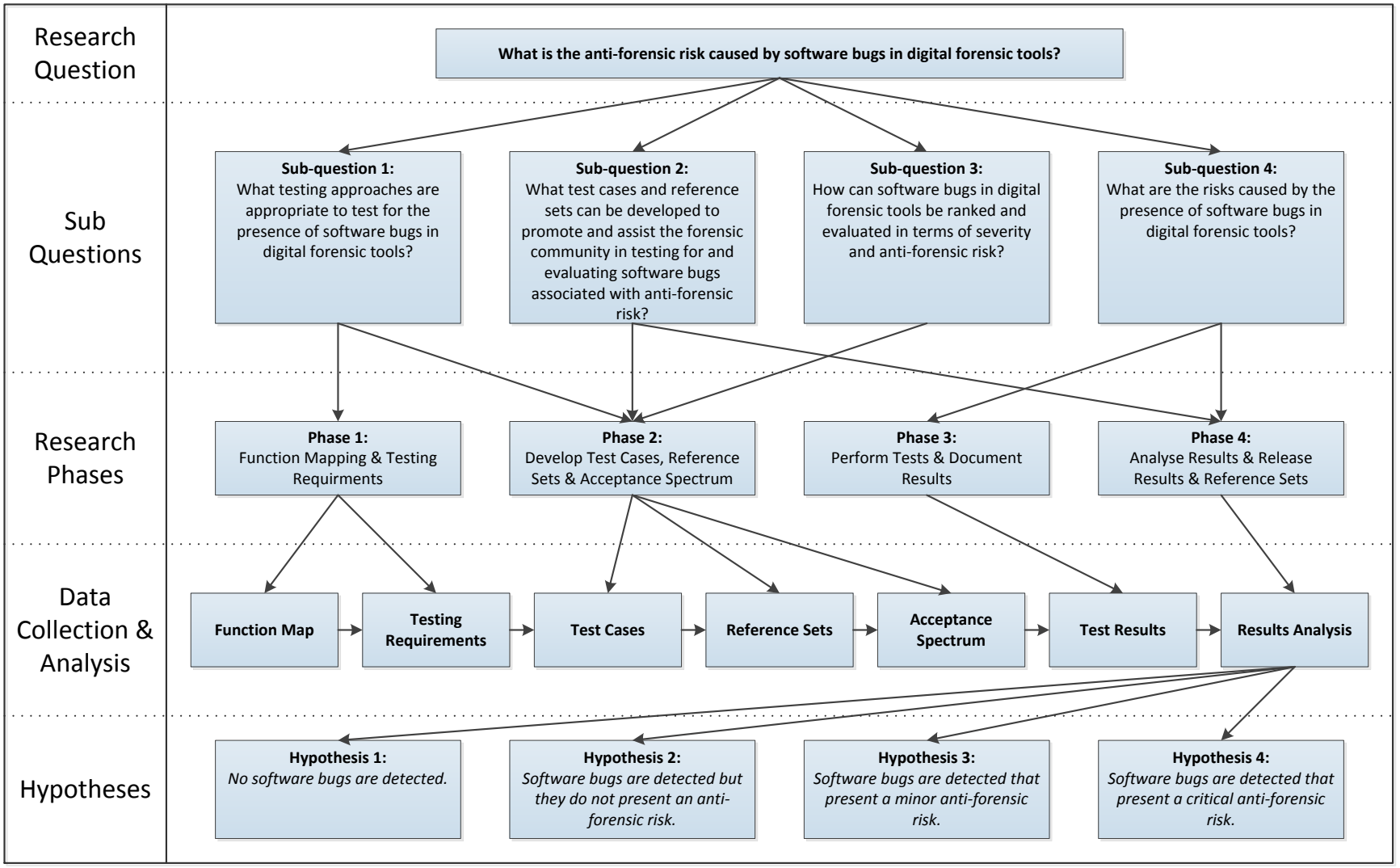

Figure 3.2: Data map

3.3 DATA REQUIREMENTS

Information will be collected in Phase 1 and Phase 2 that will allow for the development of function maps, testing requirements, test cases, test acceptance spectrums and reference sets. A series of tests will be conducted in Phase 3 based on the information collected in Phase 1 and 2. Data will be collected during the testing in Phase 3 which will be analysed and interpreted in Phase 4. The analysis will provide evidence for the testing of the hypotheses discussed in Section 3.2.3.

3.3.1 Data Collection

The various data collection methods used by the research are described in the following Sections. The data collection methods are presented in their logical order as shown on the data map in Figure 3.2. Firstly the function mapping technique is described in Section 3.3.1.1. Function mapping allows for the definition of the testing requirements described in Section 3.3.1.2 which then allows for the development of test cases as detailed in Section 3.3.1.3. Reference sets are then explained in Section 3.3.1.4. The testing methodology is described in Section 3.3.1.5 which links to the acceptance spectrums explained in Section 3.3.1.6.

3.3.1.1 Function mapping

Prior to constructing specific testing requirements, a function map is created. The function map allows for the mapping of functions and sub-functions of digital forensic tools. The function mapping process will be adapted from Guo, Slay and Beckett (2009). There are two goals of the function mapping process; firstly to identify which functions are possible candidates for testing and secondly to identify which function areas have already been extensively tested.

A number of function areas will be identified based on information from previous literature. The function area will then be mapped into individual functions and sub-functions. For example, the function area of compound containers could have function areas mapped underneath it such as file containers and compressed archives. File containers could then be further mapped into specific sub-functions such as Portable Document Format (PDF) files whereas the

compressed archives function area would have sub-functions such as a ZIP archive mapped underneath it. The ZIP archive can be mapped deeper with sub-functions such as handling nested archives and password protected archives.

Once the function areas have been appropriately mapped, it will be possible to overlay information concerning previous research and testing. For example CFTT testing has extensively tested write blocking related functions without finding any major issues; which makes this function area less suitable for testing for software bugs. In contrast, Newsham et al (2007) performed testing of MBR tables and discovered a number of issues; the implication being that MBR tables and related data structures such as GPT may be suitable for testing.

3.3.1.2 Testing requirements

Testing requirements will be derived from the function mapping process discussed in Section 3.3.1.1. The requirements will translate the functions into something that is tangible and testable. The current methodologies in use such as the CFTT methodology focus on completeness and accuracy of function. The requirements used by this research will primarily focus on anti-forensic risk, security and exception handling requirements. The implication being that this research is not concerned if a function fails to work as expected, provided there are no risks associated with the failure. For example, a document viewing function may fail to open a document; however the failure may not necessarily pose an anti-forensic risk. In contrast, if the failure to open the document results in the tool crashing and corrupting evidence files then there is a clear anti-forensic risk present.

3.3.1.3 Test cases

The testing requirements defined in Section 3.3.1.2 will be developed into full test cases. Each test case will relate to one or more testing requirements. The test requirements define what is to be tested whereas the test cases will describe how the test requirements are to be tested. The primary purpose of the test case is to describe step-by-step how the test is to be performed so that the results of testing can be reproduced. The test cases will specify details such as environmental variables like hardware and software configuration. The test cases will also specify which reference sets are to be used as input for each test.

3.3.1.4 Reference sets

A number of reference sets will be created as input for each test case. The reference sets will contain a set of malformed files designed to test a specific function or function area. Benign files will be collected and then have malformations applied to create reference sets. The primary malformation technique used will be file fuzzing; although manual targeted manipulation will also be considered where appropriate. Reference sets will be created using fuzzing at various malformation levels to investigate how digital forensic tools handle different levels of malformed input.

3.3.1.5 Testing methodology

Each test case will describe a number of functions that need to meet certain requirements. These test cases will be performed by using a number of reference sets as inputs. The testing methodology will consist of an automated process of reference sets being used as input for various functions. The output of this automated testing process will be a pass/fail result; with a pass result indicating that nothing unexpected happened. The fail results will then be manually tested and described in terms of an acceptance spectrum discussed in Section 3.3.1.6.

3.3.1.6 Acceptance spectrums

Acceptance spectrums adapted from Slay and Beckett (2007) will be used to determine the results of each test case. The acceptance spectrums contain a range of possible outcomes for each test case beyond a simple pass or fail result. The default acceptance spectrum to be used is mapped to the hypotheses discussed in Section 3.2.3 and is presented below.

Table 3.1: Default acceptance spectrum

Result	Acceptance Spectrum	Mapped Hypothesis
Pass	Exceeds expectations	H1: No software bugs are detected.
Pass	Meets expectations	H2: Software bugs are detected but they do not present an anti-forensic risk.
Fail	Unacceptable	H3: Software bugs are detected that present a minor anti-forensic risk.
Fail	Critically unacceptable	H4: Software bugs are detected that present a critical anti-forensic risk.

If necessary the default acceptance spectrum will be modified to make it more suitable for individual test cases. For example certain function areas may require that additional result ranges are defined to provide a more granular testing result.

3.3.2 Data Processing

The results of the testing will primarily be in the form of a test case and its associated result on the acceptance spectrum. There may be additional material, such as screenshots of error messages and test notes, depending on the particular test case. The results will be summarised into a table at the conclusion of each test case; the table will contain a reference to the test case, list reference sets used, test results and analysis of the results. Once all of the tests have been completed the results will be compiled into a spreadsheet to allow for data analysis.

3.3.3 Data Analysis

Analysis of the data will focus on interpreting the test results and determining the significance of the anti-forensic risks associated with any software bugs identified. The main variables of interest are the following:

- Function type;
- Input malformation level;
- Number of software bugs detected and;
- Severity of anti-forensic risks associated with software bugs.

3.3.3.1 Descriptive statistics

Descriptive statistics will be generated for each function and function area tested. This will allow for the interpretation of the results for each function area as well as the ability to make statistical comparisons between function areas. The acceptance spectrums will allow a level of granularity in the results that will allow problem areas to be identified.

3.3.3.2 Software bug analysis

Any severe or unusual software bugs will be analysed in depth through manual experimentation and analysis. This analysis will primarily consist of examining the minimum conditions needed to trigger the software bug by using different types and levels of malformed input which will be used to try and isolate the root cause of the software bug. If the root cause of the software bug can be identified then further analysis will be performed on discovering the possible consequences of the software bug.

3.4 LIMITATIONS OF THE RESEARCH

The proposed research aims to investigate the anti-forensic risks associated with software bugs in digital forensic tools. However, the research has a number of limitations. The main limitation is that the research is not able to do widespread testing across all function areas of all digital forensic tools. It is not feasible to conduct testing on a scale that would fully test all digital forensic tools; therefore a subset of functions will be targeted in a selected number of digital forensic tools.

The software and hardware in the testing environment is also a significant limitation. The testing proposed by the research will be conducted in a single stable environment. However, digital forensic tools are used in a wide range of environments. The effects of environmental variables may confound the input variables resulting in falsely attributing results to a certain reference set. For example, there may be certain test cases that fail in the testing environment that are not reproducible in other environments; similarly the testing environment may hide software bugs that are present in other environments.

The nature of software bugs also means that even with a stable testing environment, there may be certain test results that are difficult to reproduce. The difficulty in reproducing results could be caused by the complexity of the conditions required for the occurrence of the software bug.

The above limitations must be considered when interpreting the results of the research. The implication of testing a small number of function areas in a limited number of tools is that the results may not provide an accurate representation of all function areas over all digital forensic tools. For example, the field testing may not identify any software bugs in the tested function areas; however the result should not be extrapolated as evidence that all function areas are free of software bugs. The implication of difficult-to-reproduce software bugs is that a software bug identified by the research may not be reproducible by other researchers, or, conversely, that other researchers may identify software bugs where this research has identified none. The identified limitations of the research methodology used provide possible directions for further research in the area of the anti-forensic implications of software bugs in digital forensic tools.

3.5 CONCLUSION

The literature reviewed in Chapter Two and the similar studies reviewed in Section 3.1 revealed weaknesses in the established methodologies to test digital forensic tools. Current methodologies are not well suited for testing for security-related software bugs that may have associated anti-forensic risk. The current methodologies are also unsuitable for community-based testing outside of organisations with the resources to undertake formal testing. A need for an alternative approach to testing has become apparent.

The research methodology developed in Chapter Three has been designed to address the weaknesses in current testing methodologies. Research questions and hypotheses were first defined in Sections 3.2.2 and 3.2.3 and the proposed research phases were described in Section 3.2.4. The research phases included several elements of alternative methodologies that have been adapted from the similar studies reviewed in Section 3.1. These elements from alternative methodologies have been selected to counter existing weaknesses and create a suitable methodology for testing for software bugs and identifying associated anti-forensic risk. Data collection techniques such as function mapping, reference sets, acceptance spectrums and fuzzing have all been adapted from similar studies as described in detail in Section 3.3.1. Data processing and analysis techniques were then described in Sections 3.3.2 and 3.3.3. Finally limitations of the research were explored in Section 3.4.

The research methodology described in Chapter Three will guide the execution of testing of digital forensic tools. Chapter Four will present the findings of the testing as set out by the research methodology.

Chapter Four

RESEARCH FINDINGS

4.0 INTRODUCTION

The literature review in Chapter Two presented a wide range of anti-forensic risks currently faced by digital forensic tools; the most severe of these anti-forensic risks was shown to be related to software bugs. The literature review then revealed the lack of appropriate testing methodologies for identifying these anti-forensic risks. Chapter Three first reviewed a number of similar studies and then identified a number of research questions regarding the issue of software bugs causing anti-forensic risk in digital forensic tools. The research questions were then developed into a methodology for testing and identifying software bugs as related to anti-forensic risk.

Research and testing has been conducted as per the methodology described in Chapter Three. The purpose of Chapter Four is to report and summarise the findings of the field work that has been performed. The results from the field work will be reported and analysed to provide a foundation for discussion in Chapter Five. The findings from Chapter Four are intended to provide an insight into the types of issues that are present in digital forensic tools that may present an anti-forensic risk.

Chapter Four is split into four main Sections. Section 4.1 will describe the changes made to the methodology specified in Chapter Three. Section 4.2 will present a full report of the findings from the field work. Section 4.2 covers the first three phases of the methodology as described by Chapter Three. Section 4.3 covers the last phase of the methodology being the analysis stage. The next section, 4.4, summarises the research findings and analysis as graphs and tables. A conclusion of Chapter Four is then presented in Section 4.5.

4.1 CHANGES TO SPECIFIED METHODOLOGY

The testing went through the four phases as described in Chapter Three. Most of the testing was conducted according to the test methodology described in Chapter Three. However, a number of changes were necessary to progress with the data collection and analysis. Changes to the tools and functions to be tested are discussed in Section 4.1.1.

4.1.1 Tools And Functions To Be Tested

As discussed in Section 3.4 it is not feasible to conduct wide range testing of many function areas across many different digital forensic tools within the time restrictions of this thesis. Therefore the testing has been focussed on one forensic tool being EnCase Forensic and the subset of function areas included as part of the Evidence Processor functionality. EnCase Forensic has been chosen as it is one of the leading digital forensic tools available and is widely used by investigators. The Evidence Processor functionality of EnCase Forensic is new to version 7 of the software and combines a number of functions from version 6 in one place. The functions included in the Evidence Processor are focussed on automated pre-processing tasks that an investigator would typically complete before starting the more manual phases of an investigation.

4.2 FIELD FINDINGS

The field work was carried out in four phases. The first phase discussed in Sections 4.2.1 and 4.2.2 involved mapping a function area of a forensic tool and then translating this function map into test requirements. The second phase focussed on preparing for testing which involved creating reference sets and test cases. The test cases that have been created are summarised in Section 4.2.3 and the associated reference sets are summarised in Section 4.2.4. A description of the testing environment is also presented in Section 4.2.5. The third phase of field work was the execution of the test cases; the results of which are presented in Section 4.2.6. The final phase of the field work will be discussed in Section 4.3.

4.2.1 Function Mapping

The “Evidence Processor” functionality of EnCase Forensic has been mapped and a function map is presented in Figure 4.1. The purpose of the function map is to identify a number of sub-functions that are likely candidates for testing for software bugs. The function map also identifies various forms of input for each sub-function that is used to inform the creation of reference sets.

The sub-functions identified appear in black in Figure 4.1. Each sub-function then has either a specific function input in red or a function input category in yellow mapped to it. A specific function input refers to where the exact type of input is known and can be specified. A function input category refers to instances where the exact type of input is not known, there are numerous types of input or where further mapping a category would not be useful to the research. For instance, the Protected File Analysis sub-function has a function input category mapped to it as the inputs come from a third party program that was not able to be accessed during this research. In contrast, the System Info Parser sub-function was mapped to a function input category as it was difficult to identify and confirm the specific input types used by the sub-function.

From the function map in Figure 4.1, it is easy to identify clusters of inputs that might be suitable for testing. For instance, the Expand Compound Files and Find Email sub-functions both contain a variety of input types.

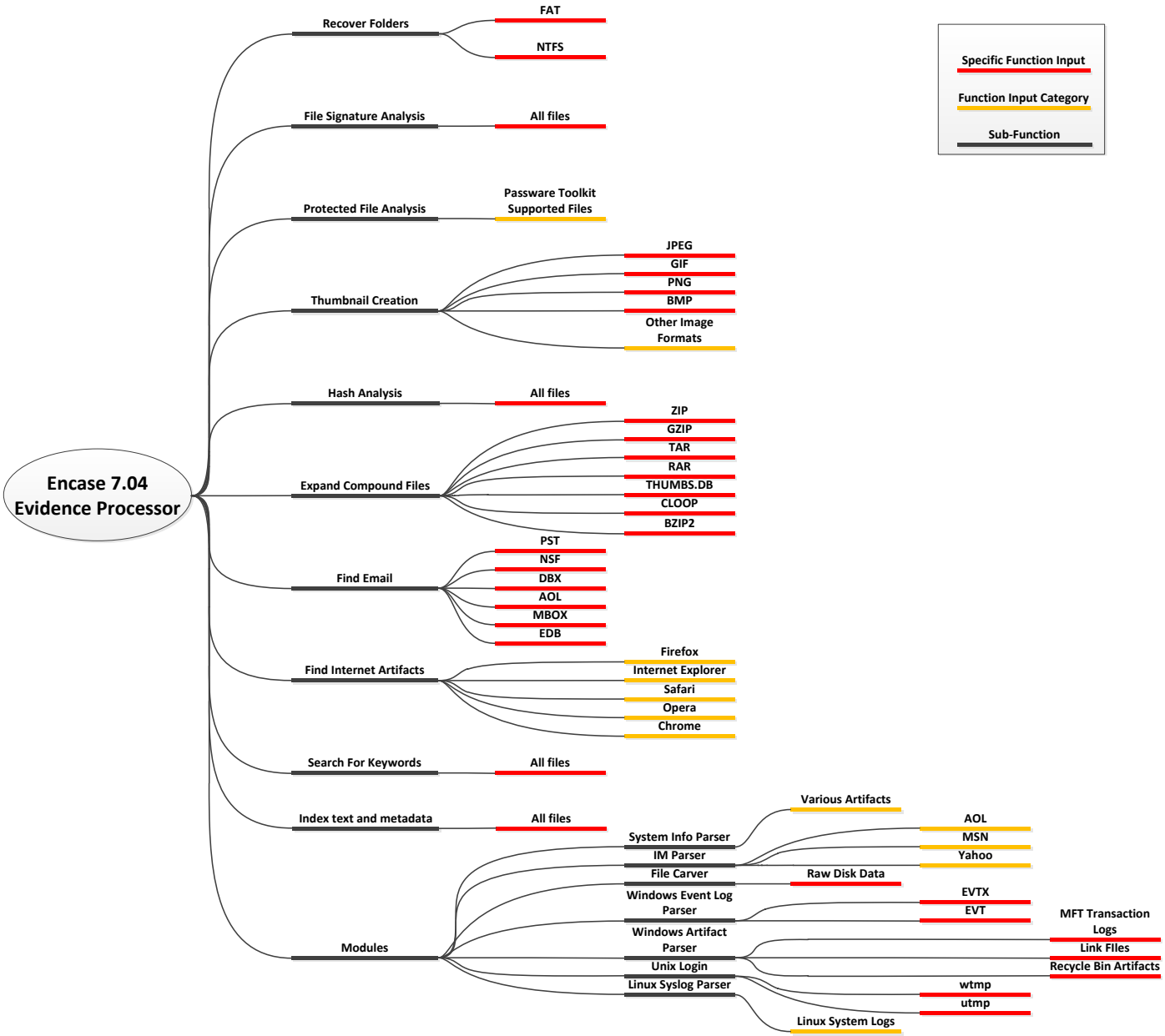

Figure 4.1: EnCase Evidence Processor function map

Table 4.1: Evidence Processor sub-functions selected for testing

Sub-Function	Selected	Reason
Recover Folders	No	Likely to be extensively tested
File Signature Analysis	No	Too simplistic (reads byte patterns)
Protected File Analysis	No	Requires third party software
Thumbnail Creation	Yes	Processes complex data structures
Hash Analysis	No	Too simplistic (mathematical function)
Expand Compound Files	Yes	Processes complex data structures
Find Email	Yes	Processes complex data structures
Find Internet Artifacts	Yes	Processes complex data structures
Search For Keywords	No	Too simplistic (reads text)
Index Text And Metadata	No	Too simplistic (reads text)
System Info Parser	No	Unable to easily identify input
IM Parser	No	Too simplistic (reads byte patterns)
File Carver	No	Too simplistic (reads byte patterns)
Windows Event Log Parser	Yes	Processes complex data structures
Windows Artifact Parser	Yes	Processes complex data structures
Unix Login	No	Too simplistic (reads text)
Linux Syslog Parser	No	Too simplistic (reads text)

In most cases sub-functions were rejected for testing because the sub-functions were deemed to be too simplistic. For example, the File Signature Analysis sub-function is too simplistic because it consists of reading a small number of bytes from a file to try and identify a file's signature. In contrast, the Expand Compound Files sub-function was selected for testing as it involves the processing of complex data structures to extract items stored inside compound

files. Generally sub-functions were selected where EnCase did some sort of processing with the input.

4.2.2 Test Requirements

Test requirements were generated based on the sub-functions selected in Section 4.2.1. A listing of the test requirements is presented in Table 4.2. The test requirements are used as the basis for creating the test cases described in Section 4.2.3.

Table 4.2: Test requirements

Requirement ID	Description
EC.EP.01	The "Thumbnail Creation" sub-function shall be able to handle malformed images without generating an error condition
EC.EP.02	The "Find Email" sub-function shall be able to handle malformed email files without generating an error condition
EC.EP.03	The "Expand Compound Files" sub-function shall be able to handle malformed compound files without generating an error condition
EC.EP.04	The "Find Internet Artifacts" sub-function shall be able to handle malformed internet artifacts without generating an error condition
EC.EP.05	The "Windows Artifact Parser" sub-function shall be able to handle malformed Windows artifacts without generating an error condition
EC.EP.06	The "Windows Event Log Parser" sub-function shall be able to handle malformed Windows event logs without generating an error condition

4.2.3 Test Cases

Six test cases have been created that are mapped to the test requirements described in Section 4.2.2. A summary of the test cases is presented below in Table 4.3. Note that the full test cases are presented in Appendix B.

Table 4.3: Summary of test cases

Test Case	Requirement	Reference Sets
TC.01	EC.EP.01	RS-IMAGE-01
TC.02	EC.EP.02	RS-EMAIL-01, RS-EMAIL-02, RS-EMAIL-03, RS-EMAIL-04
TC.03	EC.EP.03	RS-CONTAINER-01, RS-CONTAINER-02, RS-CONTAINER-03, RS-CONTAINER-04, RS-CONTAINER-05
TC.04	EC.EP.04	RS-INTERNET-01, RS-INTERNET-02, RS-INTERNET-03, RS-INTERNET-04, RS-INTERNET-05
TC.05	EC.EP.05	RS-WINDOWS-01, RS-WINDOWS-02
TC.06	EC.EP.06	RS-LOG-01, RS-LOG-02

4.2.4 Reference Set Creation

Reference sets have been created to assist with the testing process. The reference sets consist of a number of malformed files that have been created using a fuzzing process. Section 4.2.4.1 will discuss the details of the fuzzing process and Section 4.2.4.2 will present a summary of the reference sets that have been created.

4.2.4.1 File fuzzer

A simple file fuzzer was created for the purpose of generating reference sets. A simplified overview of the internal processes of the file fuzzer is presented in Figure 4.2. The C# source code for the file fuzzer is available in Appendix D.

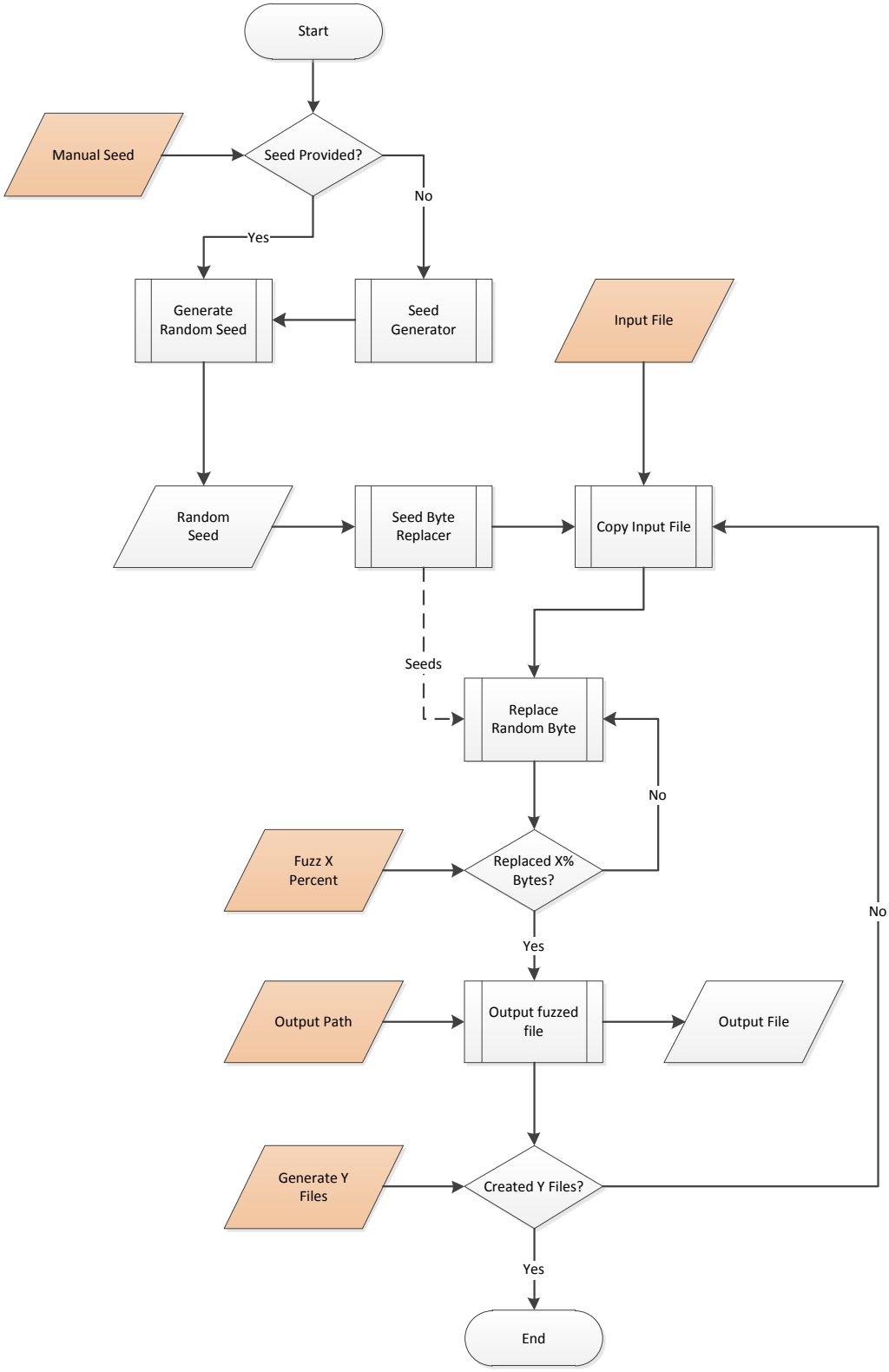

Figure 4.2: File fuzzer internal processes

The file fuzzer takes five inputs as shown in orange in Figure 4.2. The input file, output path, fuzzing percentage and number of files to generate are required inputs; however the manual seed is an optional input. The file fuzzer uses a seeded random number generator to decide which byte to replace in a file. This seeded random number generator can be provided with a manual seed from the user which will result in the same bytes being replaced each time that seed is used. If a manual seed is not provided then a random seed is generated using a secure random number generator.

The file fuzzer uses the seeded random number generator to randomly replace the specified percentage of bytes. Upon completion a malformed file is then output. The process is repeated until the specified number of files is generated.

4.2.4.2 Reference set summary

A number of reference sets have been created by first identifying a benign input file for a sub-function. A reference set is then generated by creating a number of malformed files based on the input file. The benign files were either taken from an existing image of forensic corpora or by manually creating benign files. The details of the origins of the benign files used are presented in the full details for each reference set.

The reference sets all contain various numbers of files at five specific malformation percentages being 0.1%, 0.2%, 0.5%, 1% and 2%. A range of malformation percentages was chosen to generate a wide range of possible malformations in the reference sets. One file type may generate an error condition at a malformation percentage of 0.1%; however another file type may only generate an error condition at a malformation percentage of 2%.

A summary of the reference sets created in presented below in Table 4.4. In Table 4.4 each reference set is listed by ID and a brief description of the contents is provided. The full details for each reference set are presented in Appendix A. The full details include information such as the origins of benign files and the number of malformed files generated at each malformation percentage.

Table 4.4: Summary of reference sets

Reference Set ID	Contents
RS-IMAGE-01	Malformed BMP, GIF, JEPG and PNG images
RS-EMAIL-01	Malformed PST email containers
RS-EMAIL-02	Malformed NSF email containers
RS-EMAIL-03	Malformed MBOX email containers
RS-EMAIL-04	Malformed DBX email containers
RS-CONTAINER-01	Malformed ZIP file containers
RS-CONTAINER-02	Malformed GZIP file containers
RS-CONTAINER-03	Malformed TAR file containers
RS-CONTAINER-04	Malformed RAR file containers
RS-CONTAINER-05	Malformed BZIP2 file containers
RS-INTERNET-01	Malformed Firefox history/bookmark databases
RS-INTERNET-02	Malformed Internet Explorer history databases
RS-INTERNET-03	Malformed Opera history databases
RS-INTERNET-04	Malformed Safari history databases
RS-INTERNET-05	Malformed Chrome history databases
RS-WINDOWS-01	Malformed Windows Link files
RS-WINDOWS-02	Malformed Windows Recycle Bin (INFO2) records
RS-LOG-01	Malformed Windows Legacy Event Logs
RS-LOG-02	Malformed Windows Event Logs

From Table 4.4 it can be seen that the reference set RS-IMAGE-01 is the only reference set that contains more than one type of file. RS-IMAGE-01 was the first reference set created and processing took a long time to complete. The long

processing time could cause complications when it comes to reproducing issues discovered. Subsequent reference sets were therefore split up into smaller sets.

4.2.5 Testing Environment

A stable testing environment was set up to help address the limitations discussed in Chapter Three where issues with the testing environment can confound issues with the software being tested. The complete hardware specifications for the testing workstation are shown in Table 4.5. All of the software installed on the testing workstation is listed in Table 4.6.

Table 4.5: Testing workstation – hardware specifications

Component	Model
Case	Silverstone SG08
Power Supply	Silverstone 80 PLUS Bronze Certified 600W
Motherboard	GIGABYTE GA-H61N-USB3 (BIOS Version F8)
CPU	Intel Core i7-3770 (3.4 GHz, 3.9 GHz Turbo)
RAM	8GB G.SKILL F3-10666CL9D-8GBXL
Primary Drive	40 GB Intel 320 Series SSD
Testing Drive (Type 1)	120 GB Intel 520 Series SSD
Testing Drive (Type 2)	2 TB Western Digital Caviar Black HDD
Welland eSATA Dock	EZStor ME-601J
CodeMeter USB Dongle	CMStick v1.16 (EnCase license protection)

All testing drives were sanitised prior to use through either a secure erase in the case of an SSD (Type 1) or an EnCase wipe in the case of a hard drive (Type 2). Type 2 drives were primarily used for storage of reference sets and only for testing in limited cases where the testing required more storage space than could be provided by Type 1 drives. All testing drives were all connected to the testing workstation using the Welland eSATA dock through an eSATA connection.

Table 4.6: Testing workstation – installed software

Software	Description/Purpose
Windows 7 Enterprise (64-bit)	Operating system
Various Drivers	Chipset, audio, video, storage etc.
Intel SSD Toolbox 3.03	Secure erasing SSD drives
LogMeIn 4.1.0.2450	Remote monitoring of testing
EnCase Forensic 7.04.00.85 (64-bit)	Target digital forensic tool for testing
CodeMeter for Windows 4.30.482.502	EnCase license protection

A 64-bit operating system and the 64-bit version of EnCase Forensic were selected as the use of 64-bit systems is now commonplace. One limitation of using a 64-bit operating system and software is that a lot of software testing and debugging methodology and tools are geared towards a 32-bit environment.

4.2.6 Field Findings

Field findings from the test cases that have been conducted are presented below. Results are presented individually for each of the six test cases. The results for each test case include an overview of the test case, a table summarising the results and an explanation of the results. Note that a journal of the testing process was also kept and is available in Appendix C.

4.2.6.1 TC.01: thumbnail creation

Test case TC.01 involved testing the "Thumbnail Creation" feature of EnCase's Evidence Processor. This feature creates thumbnails of all images on a suspect device to allow for the quick previewing and browsing of images. A summary of the results for each reference set associated with TC.01 is presented below in Table 4.7.

Table 4.7: TC.01 result summary

Requirement	Reference Set Tested	Result
EC.EP.01	RS-IMAGE-01	No issues

EnCase was able to successfully process the reference set RS-IMAGE-01 without any indication of software bugs or other issues.

4.2.6.2 TC.02: find email

Test case TC.02 involved testing the "Find Email" feature of EnCase's Evidence Processor. This feature identifies and parses email artifacts from a suspect device and presents them in a readable and searchable format. A summary of the results for each reference set associated with TC.02 is presented below in Table 4.8.

Table 4.8: TC.02 result summary

Requirement	Reference Set Tested	Result
EC.EP.02	RS-EMAIL-01	Crash
EC.EP.02	RS-EMAIL-02	Crash
EC.EP.02	RS-EMAIL-03	No issues
EC.EP.02	RS-EMAIL-04	Large cache files

When attempting to process either reference set RS-EMAIL-01 or RS-EMAIL-02 EnCase would crash within minutes of the starting of processing. Windows would then present the error message shown in Figure 4.3 to indicate that EnCase had stopped working. These reference sets were retested to ensure that the result was repeatable.

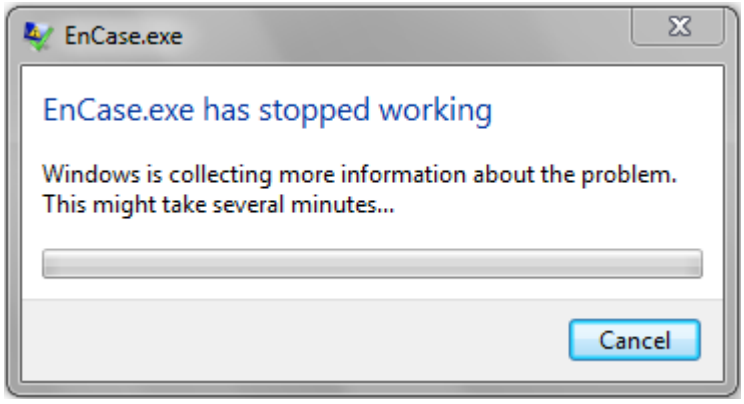

Figure 4.3: TC.02 Windows error message

When processing the RS-EMAIL-04 reference set, EnCase generated a large amount of files in the evidence cache folder. Testing was abandoned after EnCase filled a 2TB drive with cache files. The "Find Email" function in EnCase generates logical evidence files of any email artifacts it finds and stores these in

the evidence cache folder. However, in this case, the size of these logical evidence files was unusually large in contrast to the 3 GB size of the entire reference set.

4.2.6.3 TC.03: expand compound files

Test case TC.03 involved testing the "Expand Compound Files" feature of EnCase's Evidence Processor. This feature identifies and expands compound files from a suspect device to allow for searching through child items inside compound files. A summary of the results for each reference set associated with TC.03 is presented below in Table 4.9.

Table 4.9: TC.03 result summary

Requirement	Reference Set Tested	Result
EC.EP.03	RS-CONTAINER-01	Unexpected exit
EC.EP.03	RS-CONTAINER-02	Crash
EC.EP.03	RS-CONTAINER-03	No issues
EC.EP.03	RS-CONTAINER-04	Crash
EC.EP.03	RS-CONTAINER-05	Crash

While nearing the end of processing the reference set RS-CONTAINER-01, EnCase unexpectedly exited without any error message from EnCase or from Windows to indicate EnCase had crashed. A similar result was seen while nearing the end of processing the reference set RS-CONTAINER-02, however, this time, a Windows error message was shown to indicate that EnCase had crashed.

When attempting to process either reference set RS-EMAIL-01 or RS-EMAIL-02, EnCase would crash within minutes of the starting of processing. This is similar to the results seen in TC.02. These reference sets were retested to ensure that the result was repeatable.

4.2.6.4 TC.04: find internet artifacts

Test case TC.04 involved testing the "Find Internet Artifacts" feature of EnCase's Evidence Processor. This feature identifies and parses internet artifacts and presents them in a readable and searchable format. A summary of the results for each reference set associated with TC.04 is presented below in Table 4.10.

Table 4.10: TC.04 result summary

Requirement	Reference Set Tested	Result
EC.EP.04	RS-INTERNET-01	Crash
EC.EP.04	RS-INTERNET-02	No issues
EC.EP.04	RS-INTERNET-03	No issues
EC.EP.04	RS-INTERNET-04	Crash
EC.EP.04	RS-INTERNET-05	No issues

While attempting to process either reference set RS-INTERNET-01 or RS-INTERNET-04 EnCase would crash and a Windows error message would be presented. This is similar to the crashes seen in previous test cases. The processing of RS-INTERNET-03 was unusually quick and no internet artifacts were identified by EnCase. This is possibly due to the fuzzing process malforming the original internet artifact to the point where it is no longer recognisable as a valid internet artifact.

4.2.6.5 TC.05: Windows artifact parser

Test case TC.05 involved testing the "Windows Artifact Parser" feature of EnCase's Evidence Processor. This feature identifies and parses various Windows artifacts and presents them in a readable and searchable format. A summary of the results for each reference set associated with TC.05 is presented below in Table 4.11.

Table 4.11: TC.05 result summary

Requirement	Reference Set Tested	Result
EC.EP.05	RS-WINDOWS-01	No issues
EC.EP.05	RS-WINDOWS-02	No issues

EnCase was able to successfully process both reference set RS-WINDOWS-01 and RS-WINDOWS-02 without any issues.

4.2.6.6 TC.06: Windows event log parser

Test case TC.06 involved testing the "Windows Event Log Parser" feature of EnCase's Evidence Processor. This feature creates, identifies and parses Windows

event logs and presents them in a searchable format. A summary of the results for each reference set associated with TC.06 is presented below in Table 4.12.

Table 4.12: TC.06 Result Summary

Requirement	Reference Set Tested	Result
EC.EP.06	RS-LOG-01	No issues
EC.EP.06	RS-LOG-02	Internal error

Soon after the starting of processing for reference set RS-LOG-02, EnCase generates an internal error as shown in Figure 4.4. EnCase appears functional after this error has occurred but does not allow the Evidence Processor to be restarted as EnCase considers the Evidence Processor to still be running as shown in Figure 4.5. This reference set was retested to ensure that the result is repeatable.

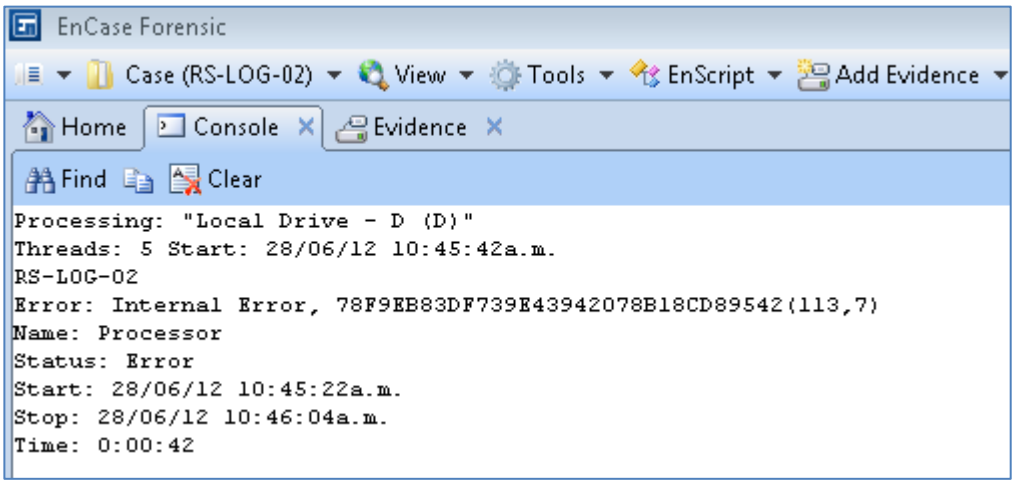

Figure 4.4: TC.06 EnCase internal error

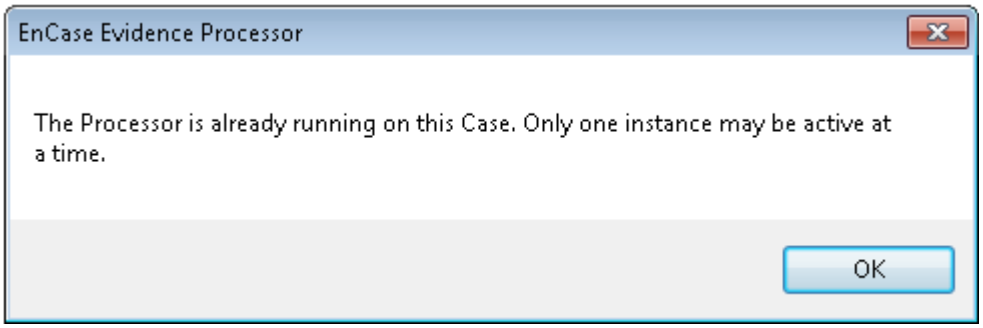

Figure 4.5: TC.06 EnCase Evidence Processor error message

4.3 RESEARCH ANALYSIS

The field findings have been reported in Section 4.2. Section 4.3 will summarise and interpret the field findings. Firstly an analysis of the issues identified is presented in Section 4.3.1. The analysis of issues will investigate the risks associated with the issues identified to better understand the significance of each type of issue. Acceptance spectrum determinations are then presented in Section 4.3.2. The acceptance spectrum determinations build upon the analysis of issues to make a determination on the results for each of the six test cases.

4.3.1 Analysis Of Issues

Throughout the testing conducted there were four distinct types of issue with EnCase that have been identified. The four types of issue are crashing, unexpected exit, internal error and the creation of large cache files. These four types of issues are described in the following sub-Sections.

4.3.1.1 Crashing

The most common type of issue seen was a complete crash of EnCase resulting in the Windows operating system presenting an error message as shown in Figure 4.3. The error message indicates that the operating system has detected a fatal exception that has occurred in the EnCase executable. Windows performs a memory dump of the application's memory and then ends the executable.

A crash has the potential to be a significant issue for an application and could result in risks such as code execution which could lead to compromising the system and evidence. Further analysis of each of the individual crashes would be needed to understand the severity and possible implications.

4.3.1.2 Unexpected exit

In test case TC.03, while processing reference set RS-CONTAINER-01 EnCase exited unexpectedly without an error message appearing from either EnCase or Windows. The lack of an error message is possibly an indication that EnCase has attempted to gracefully handle an exception but had to end the executable.

Further research would be needed to determine if the unexpected exit is the result of graceful exception handling by EnCase. It is possible that similar

risks to the crashes discussed in Section 4.3.1.1 could result from an unexpected exit.

4.3.1.3 Internal error

An EnCase internal error message occurred during test case TC.06 while processing reference set RS-LOG-02 as shown in figure 4.4. An internal error message is an indication that EnCase has encountered an exception and has been able to handle it gracefully without needing to end the executable. In this particular case EnCase remained in a working state with reduced functionality.

An internal error does not present any of the severe risks associated with a crash. The main risk of the internal error seen in TC.06 is that a large amount of evidence processing may need to be repeated.

4.3.1.4 Creation of large cache files

Testing was abandoned during test case TC.02 while processing reference set RS-EMAIL-04 due to the creation of unusually large cache files. When processing container formats, EnCase stores any processed child items into logical evidence files stored in the case cache folder. In the case of RS-EMAIL-04, the logical evidence files being created were exceptionally large. The issue seen with RS-EMAIL-04 is likely due to the manipulation of internal data structures in a DBX file by the fuzzing process. Further research would be needed to identify the exact data structures being manipulated and to test the extent to which these manipulations can disrupt processing.

The main risk of the creation of large cache files is that an investigator will run out of room to store the cache files. The end result would be that the evidence processing may need to be cancelled and repeated.

4.3.2 Acceptance Spectrum Determinations

Upon analysis of the issues identified it is now possible to determine the appropriate acceptance spectrum result for each test case as discussed in Section 3.3.1.6. The determination for each test case is presented in the following sub-Sections.

4.3.2.1 TC.01: thumbnail creation

No software bugs or error conditions were detected during the TC.01 test case and no anti-forensic risks were identified. Therefore it can be determined that TC.01 has achieved a pass result with an acceptance spectrum determination of; exceeds expectations.

4.3.2.2 TC.02: find email

Two instances of crashes and one instance of the creation of large cache files were detecting during the TC.02 test case. The software bugs detected present a minor anti-forensic risk which could prevent or disrupt the analysis of evidence. As discussed in Section 4.3.1.1 further research would be needed to determine if a crash could compromise the system or evidence and present a critical anti-forensic risk. Therefore it can be determined that TC.02 has achieved a fail result with an acceptance spectrum determination of; unacceptable.

4.3.2.3 TC.03: expand compound files

Three instances of crashes and one instance of an expected exit were detected during the TC.04 test case. The software bugs detected present a minor anti-forensic risk which could prevent or disrupt the analysis of evidence. As discussed in Section 4.3.1.1 further research would be needed to determine if a crash could compromise the system or evidence and present a critical anti-forensic risk. Therefore it can be determined that TC.03 has achieved a fail result with an acceptance spectrum determination of; unacceptable.

4.3.2.4 TC.04: find internet artifacts

Two instances of crashes were detecting during the TC.04 test case. The software bugs detected present a minor anti-forensic risk which could prevent or disrupt the analysis of evidence. As discussed in Section 4.3.1.1 further research would be needed to determine if a crash could compromise the system or evidence and present a critical anti-forensic risk. Therefore it can be determined that TC.04 has achieved a fail result with an acceptance spectrum determination of; unacceptable.

4.3.2.5 TC.05: Windows artifact parser

No software bugs or error conditions were detected during the TC.05 test case and no anti-forensic risks were identified. Therefore it can be determined that TC.05

has achieved a pass result with an acceptance spectrum determination of exceeds expectations

4.3.2.6 TC.06: Windows event log parser

One instance of an internal error was detected during the TC.06 test case. The software bug detected presents a minor anti-forensic risk which could prevent or disrupt the analysis of evidence. Therefore it can be determined that TC.06 has achieved a fail result with an acceptance spectrum determination of unacceptable.

4.4 PRESENTATION OF FINDINGS

A summary of the field findings from Section 4.2 and the analysis from Section 4.3 is presented in graphical form to assist the reader in understanding and visualising the findings.

Figure 4.6 summarises the results of each reference set that was tested and the types of issues identified. The types of issues identified are represented by the horizontal axis and the number of reference sets experiencing the issue is represented by the vertical axis.

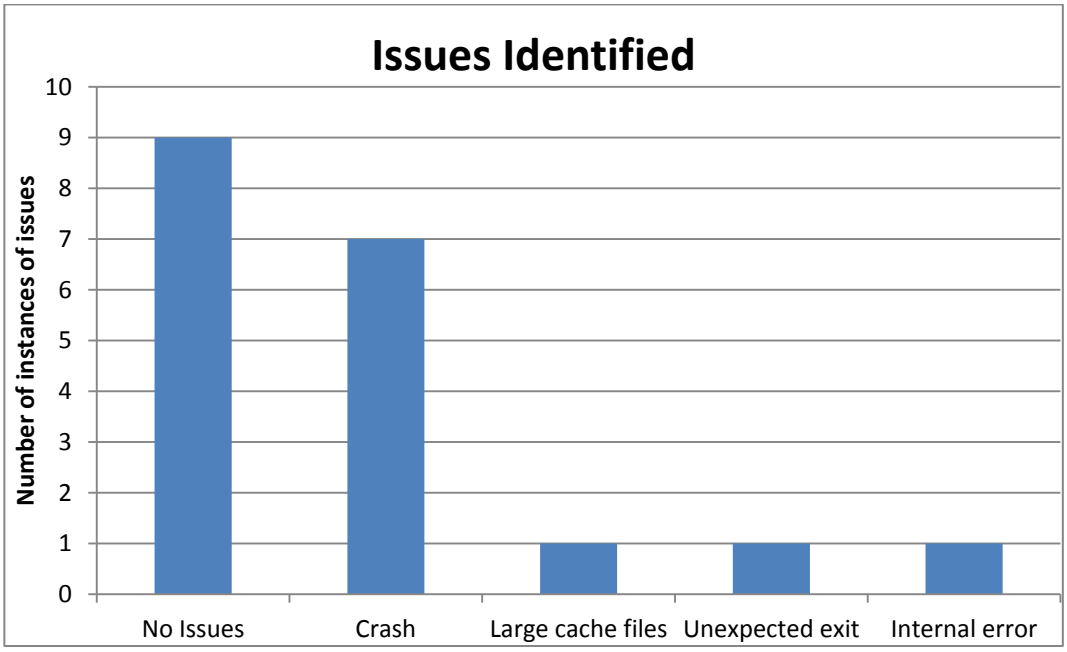

Figure 4.6: Summary of types of issues

From Figure 4.6, it is apparent that crashes were the most common type of issue identified. The other types of issues that were identified were much less common. Figure 4.6 also shows that even though there were a higher number of

crashes identified there was an even higher number of reference sets that experienced no issues at all.

Figure 4.7 provides an overview of the types of issues identified in each test case. Each individual test case is represented by the vertical axis with the issues identified in each test case depicted by different colours on the bars. The occurrence of each issue is represented on a percentage scale by the horizontal axis.

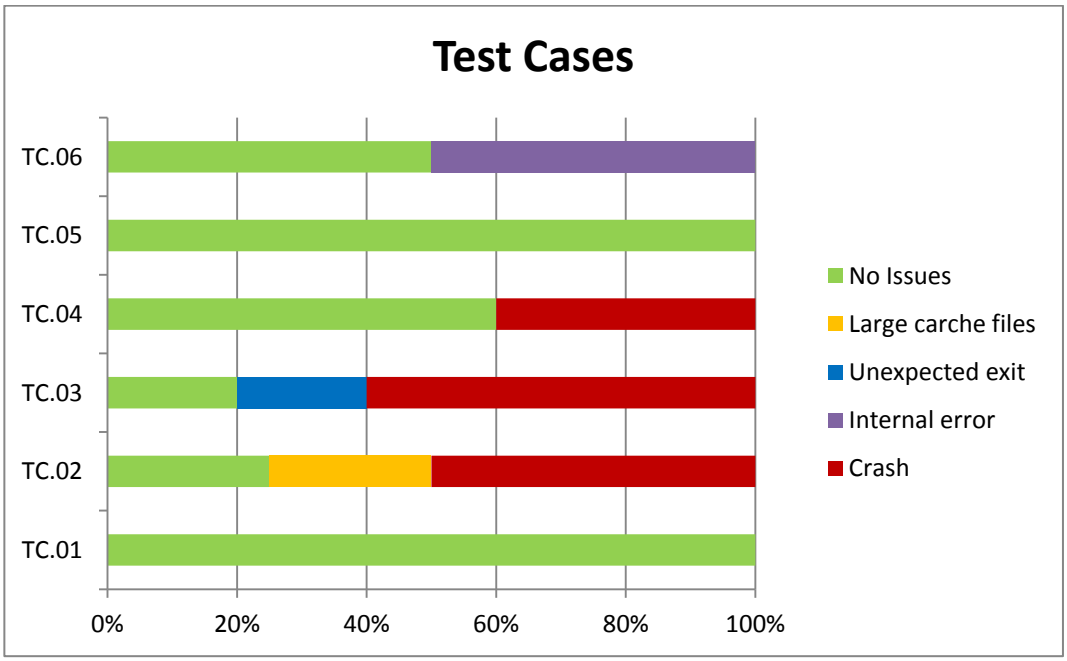

Figure 4.7: Summary of test case results

With the results broken down by test case and type of issue in Figure 4.7, it becomes apparent that the crashes experienced are clustered in three test cases; TC.02, TC.03 and TC.04. The three test cases that experienced crashes deal with complex data structures such as email containers. In contrast, the three test cases that did not experience crashes deal with simpler data structures such as images and logs. Figure 4.7 also shows that no single test case was a complete failure with at least some reference sets experiencing no issues in each test case.

Acceptance spectrum determinations for each test case were made in Section 4.2.3. Table 4.13 presents a summary of the acceptance spectrum determinations for each test case. Each test case is listed along with the associated pass/fail result and the acceptance spectrum determination.

Table 4.13: Summary of acceptance spectrum determinations

Test Case	Result	Acceptance Spectrum
TC.01	Pass	Exceeds expectations
TC.02	Fail	Unacceptable
TC.03	Fail	Unacceptable
TC.04	Fail	Unacceptable
TC.05	Pass	Exceeds expectations
TC.06	Fail	Unacceptable

Table 4.13 shows that two thirds of the test cases have resulted in an unacceptable acceptance spectrum determination and one third of the test cases have resulted in an exceeds expectation determination.

4.5 CONCLUSION

Chapter Four has reported on the variations to the methodology, field findings, analysis and presented a visual representation of the field findings. Some variations to the methodology described in Chapter Three were expected and were reported in Section 4.1. Section 4.2 then reported on the results from the field testing as well as the various data collected. The results from each of the six test cases were reported which included the reporting of a number of issues. Section 4.3 presented an analysis of the issues identified and also a determination of the acceptance spectrum results for each test case. Finally Section 4.4 summarised the field findings and presented them in a visual form.

The next chapter, Chapter Five, will discuss the field findings reported in Chapter Four. Chapter Five will present a detailed discussion of the anti-forensic implications and possible counter measures for the issues identified in the field findings. Chapter Five will also perform hypothesis testing and answer the research questions as set out in Chapter Three.

Chapter Five

DISCUSSION

5.0 INTRODUCTION

Field testing has been conducted according to the methodology described in Chapter Three. The results of the field testing are reported and presented visually in Chapter Four. Chapter Five presents a comprehensive discussion of the findings from Chapter Four to evaluate the relevance and significance of the field testing. The discussion in Chapter Five is intended to provide context for the results of the field testing in the discipline of digital forensics.

The research question concerns the anti-forensic implications arising from the presence of software bugs in digital forensic tools. Chapter Five discusses the field findings in relation to the research question. Chapter Five also provides an in-depth discussion of the field findings including discussions on the anti-forensic implications of the field findings and the effectiveness of the testing methodology used.

Chapter Five is split into four main Sections. Section 5.1 answers the research question and sub-questions developed in Section 3.2.2. The field results from Chapter Four are used to formulate an answer to the research question and sub-questions. The four hypotheses defined in Section 3.2.3 are also tested and discussed to determine the validity of each hypothesis. Section 5.2 presents a discussion of the research findings across a range of topics to provide context for the research findings in the wider discipline of digital forensics. Section 5.3 builds on the discussion of the research findings and presents recommendations for areas of further research. Section 5.4 then provides a conclusion of the discussion.

5.1 RESEARCH QUESTIONS AND HYPOTHESES

Section 3.2 presented the research question, sub-questions and associated hypotheses that are derived from the literature reviewed in Chapter Two. The research question and sub-questions will be reviewed to determine if the field findings from Chapter Four successfully answer the research question. A review of the research question is presented in Section 5.1.1 and a review of the sub-

questions in Section 5.1.2. The hypotheses from Section 3.2 will then be tested and reviewed in Section 5.1.3.

5.1.1 Research Question

The research question for this research was derived and presented in Section 3.2.2. The research question is: What is the anti-forensic risk caused by software bugs in digital forensic tools? The purpose of the research question was to better understand the severity of anti-forensic risk related to software bugs as the literature reviewed in Chapter Two did not provide a sufficient answer.

Chapter Two reviewed literature related to anti-forensic risks in digital forensic tools and an understanding of typical anti-forensic risk and testing methodologies was achieved. Chapter Three developed a methodology to answer the research question. Using this methodology, six test cases were set up and executed with the results reported in Chapter Four.

Section 4.4 presented a summary and visual representation of the results of the six test cases and also included a summary of the acceptance spectrum determinations for each test case. The answer to the research question can be investigated by reviewing the results in Section 4.4. Table 4.13 showed that two thirds of the test cases resulted in an “unacceptable” determination and one third of the test cases resulted in an “exceeds expectations” determination. None of the test cases resulted in a “meets expectations” or a “critically unacceptable” determination.

5.1.2 Sub-Questions

Four associated sub-questions outlined in Section 3.2.2 were developed to assist in answering the research question. The first sub-question is: What testing approaches are appropriate to test for the presence of software bugs in digital forensic tools? The methodology described in Chapter Three and used during the field testing for this research focussed on creating a number of reference sets containing malformed data which were then processed using digital forensic tools. The malformed data was created through a process known as file fuzzing. Figure 4.6 and Figure 4.7 shows the methodology used by the research was successful in identifying 10 different software bugs.

The second sub-question is: What test cases and reference sets can be developed to promote and assist the forensic community in testing for and evaluating software bugs associated with anti-forensic risk? The test cases and reference sets created during the research were successfully used to identify software bugs associated with anti-forensic risks. The tests cases and reference sets themselves, as well as the processes used to create them, can be used as a template by the forensic community to conduct further testing.

The third sub-question is: How can software bugs in digital forensic tools be ranked and evaluated in terms of severity and anti-forensic risk? Acceptance spectrums adapted from previous studies were used by the research to evaluate the severity of anti-forensic risks. Acceptance spectrums provide increased granularity over a simple pass/fail result which is necessary to give a better indication of the severity of anti-forensic risk. The research analysis presented in Section 4.3 also demonstrates the importance of investigating and discussing the actual impact of software bugs. While a software bug that causes a crash could potentially present a critically unacceptable anti-forensic risk, it is not appropriate to label it as such without further analysis and corresponding evidence.

The fourth sub-question is: What are the risks caused by the presence of software bugs in digital forensic tools? The research analysis presented in Section 4.3 shows that the research identified four distinct types of software bugs with associated anti-forensic risk. From the analysis performed, it is apparent that software bugs in digital forensic tools present a number of risks with the most prevalent being the risk of disrupting or preventing an investigation from taking place. The software bugs identified could potentially pose a more severe risk, such as allowing for the hiding of evidence or other, more covert methods of subverting an investigation. However it is important to note that the research has only determined this is a possible risk, not an observed risk.

5.1.3 Hypotheses Testing

Four hypotheses were developed for the research question and presented in Section 3.2.3. The four hypotheses will be checked against the field findings. Tables 5.1 – 5.4 present the four hypotheses with associated arguments for and against.

H1: No software bugs are detected.

Table 5.1: H1 testing

For	Against
Two test cases had an acceptance spectrum determination of "Exceeds expectations" which maps to H1	Ten software bugs were detected across four test cases
Conclusion: Reject	

H2: Software bugs are detected but they do not present an anti-forensic risk.

Table 5.2: H2 testing

For	Against
Ten software bugs were detected across four test cases	The software bugs detected do present an anti-forensic risk
	No test cases had an acceptance spectrum determination of "Meets Expectations" which maps to H2
Conclusion: Reject	

H3: Software bugs are detected that present a minor anti-forensic risk.

Table 5.3: H3 testing

For	Against
Ten software bugs were detected across four test cases	
Four test cases had an acceptance spectrum determination of "Unacceptable" which maps to H3	
Conclusion: Accept	

H4: Software bugs are detected that present a critical anti-forensic risk.

Table 5.4: H4 testing

For	Against
Ten software bugs were detected across four test cases	No software bugs were detected that present a critical anti-forensic risk
	No test cases had an acceptance spectrum determination of "Critically Unacceptable" which maps to H4
Conclusion: Reject	

The results indicate that H1 is not correct even though two test cases were determined to be mapped to H1. The fact that ten software bugs were detected falsifies H1. H2 is not correct as the software bugs that were detected do in fact present an anti-forensic risk. H3 is correct as the four test cases where software bugs were detected were all determined to present a minor anti-forensic risk. H4 is not correct as no software bugs were determined to present a critical anti-forensic

risk. However further research is needed to fully investigate the possibility of critical anti-forensic risks existing.

5.2 DISCUSSION OF RESEARCH FINDINGS

This Section discusses the research findings presented in Chapter Four and relates the findings to the literature reviewed in Chapter Two. Firstly a discussion of the testing methodology used by the research is presented in Section 5.2.1. The testing methodology used is compared and discussed in relation to the issues identified with existing methodologies by the literature review. Section 5.2.2 discusses the challenges involved in evaluating anti-forensic risk and the effectiveness of the strategy used by the research. Section 5.2.3 presents a critical review of the anti-forensic implications of the field findings as they relate to a practitioner undertaking an investigation. Finally Section 5.2.4 discusses how the anti-forensic risks identified could have legal implications for digital evidence.

5.2.1 Testing Methodology

Section 2.5 of Chapter Three reviewed a number of approaches for evaluating anti-forensic risk in digital forensic tools. Established methodologies such as the CFTT methodology and the SWGDE guidelines were investigated and revealed to be unsuitable for identifying and investigating software bugs related to anti-forensic risk. The iSEC Partners team noted that current testing focuses on countering data hiding techniques and ensuring accurate reproduction of evidence (Newsham, Palmer, Stamos, & Burns, 2007). The focus of current testing methodologies is not focussed on or suitable for testing for software bugs and related anti-forensic risks.

The literature review investigated testing methodologies used for software security testing in Section 2.5.4. There is a range of software security testing methodologies that are well established and proven to achieve results. However one issue with many software security testing methodologies is that they require a high level of technical expertise and knowledge in niche areas. It is unusual to find a digital forensic practitioner who also has the skill set to undertake proper software security testing. Unfortunately the niche nature of digital forensics also means it is difficult to find software security testing experts who have access to digital forensic tools and enough experience in digital forensics to understand the

implications of any issues they find. One particular software security technique, fuzzing was identified by the literature review as being potentially suitable due to its simplicity and ability to get results easily.

The literature review also investigated the digital forensic tool testing images released by Carrier (2010). The images released by Carrier are a great example of how the digital forensic community can test tools together effectively. The images provided by Carrier each test a number of simple and easy to understand functions that can be tested by anyone with a reasonable level of proficiency in digital forensics. Carrier's images focus on testing functionality much like the CFTT methodology and are therefore not very suitable when it comes to testing for software bugs related to anti-forensic risk. However, Carrier's images demonstrate that simplicity and community involvement can be successful in achieving a wider range of testing than that done by a dedicated testing organisation such as NIST.

The literature review revealed two main issues with current testing in the field of digital forensics as it relates to anti-forensic risk. Firstly, current testing almost exclusively focuses on testing the functionality of tools. Functionality is important for ensuring that evidence produced is accurate. However, functionality testing alone misses potential software bugs in digital forensic tools that could be pose anti-forensic risk. The second issue revealed by the literature review is a lack of practical testing and research regarding anti-forensic risk. Most anti-forensic research discusses theoretical risks and implications without actually doing any significant testing.

A number of similar studies were reviewed in Chapter Three and summarised in 3.2.1. The review of similar studies revealed how other researchers had attempted to improve digital forensic tool testing. Wilsdon and Slay (2006) noted that current methodologies do not satisfy the needs of the digital forensic community and suggested the need for a more simplified and streamlined methodology. Key elements of the approach suggested by Wilsdon and Slay (2006) include the use of reference sets and wider community involvement. Guo, Slay and Beckett (2009) pointed out that current testing methodologies are targeted at functions that have been decided to be important and that many areas of digital forensic tools are being left untested. The authors developed a process mapping technique to approach tool testing in a more systematic way. Newsham

et al (2007) argued that digital forensic tool testing needs to be more security focused. Current methodologies focus on testing enough use cases to ensure a function can be validated as working. Security testing requires that enough use cases are tested to conclude the function does not pose a security risk regardless of whether the function works or not. To prove their point, Newsham et al (2007) demonstrated the use of fuzzing and identified a number of security related software bugs in digital forensic tools.

Chapter Three used the implications of the literature review and similar studies to create an appropriate methodology for testing for software bugs and related anti-forensic risk. The first challenge of the methodology was deciding where to target the testing. The process mapping technique developed by Guo, Slay and Beckett (2009) was adapted for use in mapping the inputs of functions in digital forensic tools. The inputs of functions could then be judged based on their suitability for testing. One of the main goals while creating the methodology was to create something that was simple and easy for the digital forensic community to replicate. Because of the need for simplicity fuzzing was chosen as the main method of testing for software bugs. A custom fuzzer was created that was able to generate a number of reference sets containing malformed data that could then be input into specific functions of digital forensic tools in an attempt to locate software bugs. The malformed reference sets adapted the idea of a wider range of reference sets as proposed by researchers such as and Wilsdon and Slay (2006). The malformed reference sets are designed to be the basis of widespread community testing in a way similar to that of the images created by Carrier (2010). The difference between Carrier's images and the malformed reference sets is that the focus is on identifying software bugs, not on testing functionality.

The testing methodology that was developed was then successfully used to identify a number of software bugs in a digital forensic tool. The field findings in Chapter Four show that the malformed files in the reference sets caused various crashes, errors and other issues to occur in EnCase Forensic. The most common type of issue identified was that the reference set would cause EnCase Forensic to crash. However, other issues such as causing EnCase Forensic to produce large cache files and causing internal errors were also identified. The field findings from Chapter Four demonstrate that the testing methodology works and can be used to successfully identify software bugs in digital forensic tools. The testing

methodology presents a possible alternative to traditional function-based testing of digital forensic tools. Importantly, the testing methodology is simple to understand and has been proven to be successful in identifying software bugs.

5.2.2 Evaluating Anti-Forensic Risk

Section 2.3 of the literature review in Chapter Two investigated the range of different types of anti-forensic risk that exist as well as the various factors that contribute to this risk. Section 2.4 then reviewed past and existing anti-forensic risks related to EnCase Forensic. The literature review revealed that anti-forensic risk is caused by a wide range of factors and comes in various levels of severity. A key issue in developing the testing methodology was determining some way of determining the type and severity of anti-forensic risk associated with any software bugs identified. As shown by the field findings in Chapter Four software bugs were identified in EnCase Forensic. However, a significant challenge is determining what the actual anti-forensic implications of these software bugs are.

The literature review showed that traditional digital forensic tool testing focused on testing functionality and presenting a pass/fail result. The review of similar studies in Chapter Three revealed that Slay and Beckett (2007) had developed the concept of acceptance spectrums for providing increased granularity in determining the outcome of testing results. The acceptance spectrum concept was adopted into the testing methodology used. Section 4.3.2 shows how acceptance spectrums were applied to the test cases to determine the severity of anti-forensic risk. The acceptance spectrums informed the hypothesis testing done in Section 5.1.3. The use of acceptance spectrums ensures that anti-forensic risks are not incorrectly judged to be harmless or critical due to there being no middle ground. Hypothesis testing revealed that the majority of the test cases performed were determined to be unacceptable. Essentially, the software bugs detected have been determined to fall in the middle of the acceptance spectrum and present a minor anti-forensic risk. The result is good from the point of view that the software bugs have not been incorrectly labelled as harmless or critical. However, the result is ultimately unsatisfying in determining the actual anti-forensic implications. The acceptance spectrum result does nothing to inform investigators on whether they should be concerned about a particular software bug.

For example, one of the bugs could be analysed further and developed into a code execution vulnerability while the rest of the bugs could remain minor risks. The problem with using scales like acceptance spectrums is that it can only represent a risk's severity based on what is known about the risk. The acceptance spectrum does not take into account the possible risks that could exist. When evaluating anti-forensic risks, examiners should not necessarily take a result at face value. Care should be taken to understand if a risk has been accurately represented or whether further analysis could determine whether the risk has been under or over stated.

5.2.3 Anti-Forensic Implications And Counter Measures

The previous section discussed the issues in evaluating anti-forensic risk. The acceptance spectrum determinations in Section 4.3.2 and the hypothesis testing done in Section 5.1.3 reveal the majority of test cases align with hypothesis H3. Hypothesis H3 states that the software bugs detected present a minor anti-forensic risk. However, the acceptance spectrums and hypothesis testing only present a very high level overview of the actual anti-forensic implications of the software bugs identified. Section 4.3.1 provided a summary of the four types of issues that were identified and the possible anti-forensic implications.

The most prevalent issue identified was seven instances of crashing where EnCase suffered some sort of fatal error that it could not recover from. Two instances of similar issues, internal errors and exiting unexpectedly, were also identified, where EnCase handled the errors more gracefully. Section 4.3.1.1 mentioned that the crashes identified could potentially be developed into code execution vulnerabilities. A code execution vulnerability would present a severe anti-forensic risk; however, the risk is merely hypothetical at this stage and would require further investigation. While it is true that a crash is usually a first step towards to developing a code execution vulnerability, many crashes are mostly harmless. The demonstrable implication of the crashes identified, is that the Evidence Processor in EnCase is going to stop in the middle of processing. The typical use of the Evidence Processor is to get a large amount of automated pre-processing out of the way before conducting a manual analysis. For example, an investigator would use the Evidence Processor to expand all the compound files

from an exhibit before running a keyword search. Section 4.2.6.3 shows that multiple software bugs resulting in a crash have been identified in the "Expand Compound Files" feature of the Evidence Processor.

A suspect, knowing they are about to be investigated, and knowing the basics of the forensic process, might deliberately plant several malformed compound files on their computer. An investigator could spend a considerable amount of time running the Evidence Processor only to encounter a deliberately malformed compound file that causes EnCase to crash. The investigator has now lost a significant amount of processing time and has been prevented from using the Evidence Processor functionality to automatically expand compound files. If the investigator was to try using the Evidence Processor again they would encounter another crash and lose even more time.

At this stage the investigator is forced to take counter measures to handle the malformed compound files. The simplest option would be to manually expand the compound files one by one so that less work is lost if a malformed compound file is encountered. Another approach is to use the debugging features of the software and work with the software vendor to identify and fix the software bug. As an immediate solution, the software vendor could possibly assist in identifying the problem files so that they can be isolated. A third option is to use an alternative tool to perform the analysis on the exhibit with the hope that different tools will have different software bugs.

By analysing the crash scenario presented in the previous paragraphs we can identify a number of different types of risk factors present. The most obvious risk factor is tool risk that was first discussed in Section 2.2.2.2. The crashes are due to what Garfinkel (2007) calls a failure to validate data. The digital forensic tool has failed to validate the data it is processing from the compound file and this has resulted in a fatal error occurring. There is also an element of process risk as discussed in Section 2.2.2.3. Rogers (2005) warns that reliance on standardised processes makes it easier to target anti-forensic attacks. Expanding compound files is going to be part of many people's forensic process and therefore it becomes a promising function to target with an anti-forensic attack. A number of counter measures were presented to the above scenario; however these counter measures are dependent on human risk factors such as those discussed in Section 2.2.2.1. If an investigator is inexperienced, they are going to be less successful at

coming up with and executing a counter measure. An investigator who had not encountered similar issues before might simply keep rerunning the Evidence Processor and expect it to work the second time.

The software bugs identified that caused a crash, unexpected exit or internal error present an anti-forensic risk that is easily identifiable by an investigator. When EnCase crashes in the middle of running the Evidence Processor it is obvious that something has gone wrong and some action needs be taken to remedy the situation. There was one software bug identified that is not as obvious. Section 4.2.6.2 shows a bug with the "Find Email" function of the Evidence Processor in EnCase that results in the creation of large cache files. The "Find Email" function parses email container files and extracts the individual emails out to logical evidence files to allow further analysis. A software bug was identified with the processing of DBX email containers which resulted in unusually large logical evidence files being created. In the test case the reference set of 3GB of DBX email containers expanded to full a 1.8TB drives before the process was cancelled after 24 hours of processing time.

If such files were to be encountered in actual investigation, there would be no indication of what had happened until they run out of available hard drive space. The anti-forensic implications are similar to a crash in that the investigator loses a significant amount of time. However the loss of time is potentially much larger as the investigator is not immediately notified that something has gone wrong. The investigator must also spend additional time diagnosing the issue to figure out that EnCase has created large cache files and cleaning up the mess. The investigator can use similar counter measures to those used to counter a crash, however, there may be problems identifying the exact issue. The software bug does not cause EnCase to crash and the Evidence Processor is able to somewhat successfully parse the DBX files. The creation of large cache files is tool risk and falls into the category of denial of service attacks as defined by Garfinkel (2007) and discussed in Section 2.2.2.2. The investigator is being prevented from using the resource of hard drive space in a subtle manner without the issue becoming immediately apparent.

The primary anti-forensic implication of the software bugs identified is that investigators could lose a significant amount of time on an investigation. The software bugs identified do not currently pose a significant threat to evidential

integrity or the security of examiner computers. Although there may be potential for more severe anti-forensic risk, the only demonstrable risk is that of disrupting or preventing investigations from occurring. There are viable counter measures available to the anti-forensic risks identified; however, these may be expensive in terms of examination time and the cost of additional resources and tools.

5.2.4 Evidential Implications

The literature reviewed in Chapter Two examined some of the evidential challenges and implications associated with anti-forensic risk. Carrier (2002, p. 1) notes that "To date, there have been few legal challenges to digital evidence, but as the field matures this will likely change". There has been little challenge to digital evidence with even less attention to challenges associated with anti-forensic risk. There are several ways that anti-forensic risks could impact on the evidential value of digital evidence.

The most immediately apparent way to challenge digital evidence using anti-forensic risk is to use the core concepts of evidence and expert witness testimony. Section 2.4.1 reviewed the nature of digital evidence and its requirements for acceptance in the courtroom. In US courts, the Daubert Standard and Rule 702 of the Federal Rules of Evidence are two of the main guidelines for deciding whether evidence can be accepted or not. The Daubert Standard has a list of five recommended guidelines when determining the acceptability of evidence from an expert witness:

- Testability – Can the theory or technique used by tested? Can the theory or technique be refuted or falsified?
- Peer review – Has the theory or technique been subject to peer review or publication?
- Error rate – What is the known or potential error rate for the technique used?
- Standards and controls – Do any standards or controls exist relating to the technique used? How well are the standards or controls maintained?
- Accepted by scientific community – Does the relevant scientific or industry community generally accept the technique used?

All of the guidelines refer to the theory or technique being used. In terms of digital evidence, this could be, for example, the technique used to manipulate a data structure such as the Windows registry to extract useful evidence. Even a serious anti-forensic risk such as arbitrary code execution would not necessarily invalidate the actual technique being used. It is possible that an anti-forensic risk could use a software bug that also demonstrated a fundamental issue with the technique. In many cases it is likely that anti-forensic risks are related to issues which could affect guidelines such as the testability and the error rates of techniques. However it is important to note that the anti-forensic risks themselves do not necessarily challenge these guidelines. If a large number of software bugs with related anti-forensic risk were exposed in a digital forensic tool the legal focus would be on the software bugs and not the anti-forensic risk. The software bugs identified and discussed in Chapter Four could potentially be related to issues with the underlying techniques used to collect digital evidence. For example, the software bug discussed in Section 4.2.6.2 relates to the processing of DBX files. The software bug likely exists due to a flawed understanding of the DBX data structure or a flawed implementation of a technique to process the DBX data structure. The flaws in the technique could potentially impact on the accuracy of the technique being used and therefore pose a challenge to the Daubert guidelines. However, the anti-forensic risk of creating large cache files associated with the software bug does not necessarily challenge any of the Daubert guidelines itself.

A key concept in digital forensics is authenticity, which was first discussed in Section 2.4.2. Digital forensic investigators have well established methods to ensure that evidence they collect can be proven to be authentic. In order for evidence to be considered authentic the evidence presented (e.g. the forensic image) must be shown to be the same as that original exhibit that was collected (e.g. the suspect's hard drive). Authenticity is typically proven by the use of extensive chain of custody documentation that shows how unlikely it is that the evidence has been tampered with (Ridder, 2007, pp. 4-5). The consensus amongst digital forensic investigators and law expects such as Ridder (2007, p. 5) is that evidence obtained from forensic images satisfies the authenticity requirement. However, in his follow-up paper to the anti-forensic risks demonstrated by Newsham et al (2007) Ridder (2007, p. 9) comments that forensic images could

be found to be inauthentic due to vulnerabilities and the possibility that the forensic images have been altered. Ridder (2007, p. 8) also mentions that the authenticity of core concepts in digital forensics such evidence hash values could be challenged if risks like code execution vulnerabilities can be shown to be present in forensic software. A code execution vulnerability could potentially allow someone to compromise the investigators system which would allow them to alter evidence or at least alter what is presented to the investigator. Section 4.3.1 presented an overview of the types of issues identified and the anti-forensic implications of these issues were discussed further in Section 5.2.3. The software bugs identified do not have any demonstrable anti-forensic risks associated with them as severe as code execution vulnerabilities. The anti-forensic implications of the software bugs identified are limited to disrupting the investigation process and do not pose a challenge to the concept of evidence authenticity.

Section 2.4.3 discussed the concept of reliability of digital forensic tools. The digital forensic community has put a lot of effort into ensuring the reliability of digital forensic tools including testing by organisations such as NIST. Ridder (2007, pp. 8-9) notes that current tool testing is extensive, performs the processes it claims to and produces valid and accurate results. However, current testing does not look for software bugs that could result in security vulnerabilities. Ayers (2009) discovered several flaws in EnCases handling of times and dates. Newsham et al (2007) demonstrated a number of software bugs with associated anti-forensic risks. When presented with challenges to tool reliability such as those discovered by Ayers (2009) and Newsham et al (2007) the typical response from vendors is that a properly-trained and certified investigator should be able to work around the issue. However, anti-forensic risks may not always be immediately apparent to the investigator to allow them to find a workaround. Ridder (2007, p. 8) agrees and notes that vulnerabilities in forensic software may not be detectable by the examiner and also suggests that certification standards for examiners are not a sufficient defence against anti-forensic risk. Section 5.2.3 discussed the anti-forensic implications of the software bugs identified and included an anti-forensic risk involving the creation of large cache files that is not immediately apparent to the investigator. An investigator is not always able to determine if an anti-forensic risk has occurred and neither are they always able to determine how severe that risk is. Many of the software bugs identified resulted in

a crash which could be an indication of a minor fault or it could be an indication of a code execution vulnerability being exploited by an anti-forensic attack. Essentially end users of digital forensic tools are limited in how accurately they can judge the reliability of the tools they use. Vendors need to be more transparent and vigilant in their response to issues that could affect the reliability of their products. Ridder (2007, p. 8) has proposed the use of strict industry wide security standards for digital forensic tools and Ayers (2009) has suggested vendors should be forced to alert the forensic community about flaws that affect the reliability of forensic tools.

Section 2.4.4 discussed the hypothetical situation of an investigator using the possibility of anti-forensic risk to create reasonable doubt about a suspect. As noted by Brenner, Carrier, and Henninger, (2005) the confusing of jurors with complex technology can result in jurors finding reasonable doubt when there should be none. Unfortunately the biggest evidential implication of anti-forensic risk is not going to come from tangible and credible challenges to evidence guidelines, authenticity or reliability of tools. There is the possibility that in a legal dispute both sides may attempt to use fear and misunderstanding to exaggerate the severity of an anti-forensic risk. A precedent for this kind of legal strategy was seen in the case of Kevin Mitnick where a judge was convinced Mitnick could start a nuclear war by whistling into a payphone (Mills, 2008). It is not difficult to see how similar misunderstandings could arise regarding the ability of a suspect to interfere with a forensic examination. The software bugs identified by this research do have some demonstrated anti-forensic risk as well as the potential for severe anti-forensic risk that cannot yet be demonstrated. However it is important that when investigating such anti-forensic risk, researchers do not overstate or exaggerate the actual implications. If anti-forensic risk becomes a topic of debate in the court room the focus should be the actual demonstrable implications of any issues with digital forensic tools and not hypothetical risks.

5.3 RECOMMENDATIONS FOR FURTHER RESEARCH

The testing and field findings have been successful in answering the research questions. However, the discussion of the findings shows that there are still many

related areas of research that could be investigated. Various areas for further research are discussed in the following Sections.

5.3.1 Alternative Testing Methodologies

Section 2.5 reviewed existing testing methodologies used in the digital forensic community and revealed a number of weaknesses present in existing methodologies. Current testing focuses on function-based testing of a limited subset of functions in digital forensic tools. Function-based testing is good from the point of view of ensuring that digital forensic tools are producing reliable and accurate evidence. Function-based testing is also somewhat successful in countering traditional anti-forensic risks such as data hiding. However, function-based testing is inadequate for testing which is not necessarily dependent on a function working as expected, such as testing for security issues related to software bugs. The digital forensic community should continue to use existing testing methodologies such as the CFTT methodology and the SWGDE guidelines. However, there is also the need to develop alternative methodologies to fill in the gaps in the traditional testing methodologies. This research successfully used an alternative methodology composed of techniques such as fuzzing and the use of function mapping and acceptance spectrums. But it is important to note that alternative testing techniques are still in their infancy and need to be developed and formalised further.

5.3.2 Targeted Fuzzing

The technique of fuzzing was used by the research to successfully identify a number of software bugs with associated anti-forensic risk. Fuzzing was used in a blind manner to randomly change bytes within a file. With understanding of the underlying data structure of a file, it is possible to intelligently target specific areas of a file with specific changes. Targeted intelligent fuzzing could greatly improve the efficiency of fuzzing compared to blindly changing bytes. Using fuzzing in a more sophisticated manner has been done by the security community for some time now and these techniques could be adopted and adapted by the digital forensic community. The only prerequisite knowledge for being able to target fuzzing is to know how a specific data structure works in order to determine

what data is and is not expected by tools interpreting the data. Fortunately digital forensic practitioners already have in depth knowledge about many data structures. For example, a digital forensic practitioner is likely to have a good knowledge about data structures such as partition tables and file systems. Combining the existing knowledge of the digital forensic community with targeted fuzzing could allow for the testing of areas of digital forensic tools not possible with traditional testing techniques.

5.3.3 In Depth Analysis Of Software Bugs

A number of software bugs have been identified by the research as presenting an anti-forensic risk. However, further analysis is required to determine the nature of the bugs and the extent of the anti-forensic risk. There are also many other software bugs reported in digital forensic tools where little investigation has been performed to determine associated anti-forensic risk. Further research involving in-depth analysis of software bugs in digital forensic tools is required. However, detailed knowledge of areas such as reverse engineering, debugging and security vulnerabilities is required to perform further analysis of software bugs. There are a large number of research possibilities in this area for someone with the right skills.

5.3.4 Community Testing Framework

Carrier (2010) has created a number of sample images and distributed them to the community with the hope that the digital forensic community can use them to perform testing without relying on vendors or organisations such as NIST. Carrier provides a sample image and details the expected results from testing. Anyone from the community is able to use Carrier's resources, perform testing and then return the results to Carrier. The advantage of community-based testing, such as that implemented by Carrier, is that it allows for the testing of tools across a wide range of environments which is something that is difficult for a single organisation to do. A community-based testing framework should be set up in a similar manner to that implemented by Carrier. The first challenge in implementing community testing is getting the resources to develop and maintain the framework. Authoritative members of the digital forensic community

including academic researchers, practitioners, testing organisations and vendors would need to come together in order to support a community testing framework. The framework needs to allow for the easy distribution of singular test units to the digital forensic community. Research is needed to determine how to structure and standardise test units and how to best distribute them. Another challenge is in determining how to receive testing results back and evaluate the results. A community testing framework would be a large undertaking but would provide an invaluable resource to the digital forensic community.

5.3.5 Automated Reference Set Creation

Creation of testing materials including input data, test requirements, test methodology and expected outcomes all takes a considerable amount of effort and time. The research used fuzzing in order to simplify the creation of reference sets containing large numbers of malformed files. Although techniques such as fuzzing help with the creation of input data there is a need for improved techniques for automating the creation of reference sets as a whole. Wilsdon and Slay (2006) proposed that an automated tool be created that could that could dynamically create a disk image with content designed to test specific functionalities of a tool as selected by the user. The automated tool would output a disk image as well as a report with test cases detailing which functions the reference sets test. Importantly Wilsdon and Slay (2006) also suggest that the tool produces a unique identifier for each reference set. The importance of the unique identifier is that it allows the community to distribute and refer to the identifier rather than the reference set which may be large in size. Ideally an automated reference set generator would be integrated into a community testing framework as discussed in Section 5.3.4. There are many challenges involved with the creation of an automated tool, the most significant of which is developing algorithms to dynamically create input data.

5.3.6 Automated Testing

Currently most testing of digital forensic tools is done in a largely manual manner. A tester will manually input some data into a tool and perform a function and then observe the results. It is possible that much of the testing of digital forensic tools

can be automated. The software engineering community has spent decades developing techniques to automate testing of software which can likely be adapted and used by the digital forensic community. A major challenge in automating testing of digital forensic tools is that the majority of tools are proprietary black boxes that are not intended to be used in an entirely automated manner. There are some exceptions such as the open source Sleuth Kit, where traditional white box testing techniques such a unit testing can be used for validation. Some digital forensic tools such as EnCase provide some ability for automation through internal scripting using EnScript. Where tools do not allow for scripting, it is likely possible to automate testing using external tools such as AutoIt. Research is needed to develop automation techniques to speed up the testing process and remove the requirement for manual interaction. For example, if a tester is testing to see whether a specific tool can correctly identify all the partitions in a disk image, there is no reason why this task cannot be fully automated. An automated script could be created to load the disk image into the tool and interpret the results returned from the tool. If automated testing could be coupled with automatic reference set generation, as discussed in Section 5.3.5, there is the possibility for an entirely automated end-to-end testing process. However there are significant issues to overcome in reaching this goal, including the requirement to develop and validate automated testing frameworks for each function of each tool. Another significant challenge is maintaining the automated testing frameworks. With every update to a tool there may be the need to update and revalidate the automated testing framework.

5.4 CONCLUSION

A comprehensive understanding of the field findings from Chapter Four has been formed through discussion in Chapter Five. This chapter places the field findings into context by discussing the relevance of the field findings to the wider discipline and associated literature. The discussion of the field findings has critically reviewed the testing and anti-forensic risk evaluation methodologies used. The relative success of the methodology used and its associated strengths and weaknesses were examined. The anti-forensic implications of the software bugs reported in the field findings were discussed and a thorough understanding

was established. The evidential implications of the field findings were also explored.

The research question and sub-questions were answered in Sections 5.1.1 and 5.1.2 based on the field findings from Chapter Four. The four hypotheses were then tested in Section 5.1.3 by developing arguments for and against each hypothesis. The hypothesis testing revealed that H1, H2 and H4 are not correct while H3 was shown to be correct.

The discussion of the field findings also revealed that many areas relating to software bugs and associated anti-forensic risk require further research and investigation. Based on the discussion of the field findings, a number of recommendations for areas of further research were explored. The recommendations focused on methods to improve the testing of digital forensic tools as they apply to anti-forensic risk.

Chapter Six summarises and concludes the research project. The research findings from Chapter Four and the significant issues from Chapter Five will be summarised. The limitations of the research will also be outlined and discussed. Finally, the discussion of recommended areas for further research will be summarised to provide a link to further research.

Chapter Six

CONCLUSION

6.0 INTRODUCTION

The research has evaluated a digital forensic tool for the presence of software bugs and any associated anti-forensic risk. A literature review and a review of similar studies formed the basis for the development of a methodology for testing for software bugs with associated anti-forensic risk. The methodology was then used to perform field testing of a digital forensic tool. The subsequent results and discussion of the field testing reveal the presence of software bugs that do pose an anti-forensic risk.

The literature review in Chapter Two identified the importance of research into the area of software bugs and related anti-forensic risk. Digital forensic practitioners typically rely on a small number of large and complex software packages to perform forensic analysis. Digital forensic tools are relied on to provide evidence relating to serious issues and are universally accepted by court rooms around the world. The literature examined the various types of anti-forensic risk and presented a case study of past risks that have been shown to exist in forensic tools such as EnCase. The literature review also examined current testing techniques and revealed that current testing of digital forensic tools fails to address the presence of potential anti-forensic risk. The potential consequence of a software bug that poses an anti-forensic risk may be a malicious person being able to hide or alter evidence or disrupt or prevent an investigation

Chapter Six will present a final conclusion to the research. The field testing results and discussion from Chapters Four and Five are summarised in Section 6.1. The answer to the research question is then summarised and concluded in Section 6.2. Limitations of the field testing and research are discussed in Section 6.3. Recommendations for further research based on the discussion in Section 5.3 are summarised in Section 6.4. Finally Section 6.5 provides a final conclusion to the thesis.

6.1 SUMMARY OF RESEARCH FINDINGS

The field testing consisted of first performing function mapping of the digital forensic tool EnCase to identify function areas that would be suitable for testing for software bugs with associated anti-forensic risk. Using the function map, six function areas were identified as being suitable. Six test cases were developed based on the chosen function area. Each test case had a number of malformed reference sets associated with it to be used as input in the field testing. The result for each test case was determined with use of acceptance spectrum determinations.

During the field testing, four distinct types of issues were observed. The four types of issues were crashing, unexpected exit, internal error and the creation of large cache files. The most common issue observed was a complete crash of EnCase resulting in the Windows operating system presenting an error message. The error message indicates that the operating system has detected a fatal exception occurring in the EnCase executable. A crash has the potential to be a significant issue for an application and could result in risks such as code execution which could lead to compromising the system and the evidence. One instance of an unexpected exit was observed where EnCase exited unexpectedly without an error message appearing from either EnCase or Windows. The lack of an error message is possibly an indication that EnCase has attempted to gracefully handle an exception but had to end the executable. One instance of an internal error in EnCase was observed. An internal error message is an indication that EnCase has encountered an exception and has been able to handle it gracefully without needing to end the executable. In this particular case, EnCase remained in a working state with reduced functionality. Testing was also abandoned in one instance due to the creation of unusually large cache files. EnCase produces cache normally under a number of circumstances but in this particular case, the cache files were unusually and unexpectedly large.

The four issues observed were present across four of the six test cases. The four test cases with issues were determined to have an acceptance spectrum result of “unacceptable”. Crashes were observed in three of the four test cases with issues; however, further research is needed to determine the extent of the anti-forensic risk present. Crashes have the potential to present critically unacceptable anti-forensic risk; however, the research has not confirmed the presence of such

risks. Two out of the six test cases were determined to have an “exceeds expectations” acceptance spectrum determination as no software bugs or error conditions were observed during testing.

6.2 ANSWER TO THE RESEARCH QUESTION

A research question and four sub-questions were derived from the reviewed literature to guide the research. The research question is: What is the anti-forensic risk caused by software bugs in digital forensic tools? The purpose of the research question was to better understand the severity of anti-forensic risk related to software bugs. A methodology was established to answer the research question that included six test cases. Four out of those six test cases resulted in an acceptance spectrum determination of “unacceptable”, while two resulted in an “exceeds expectations” acceptance spectrum determination. No test cases could be demonstrated to be “critically unacceptable” despite the observation of a large number of crashes. From initial analysis of the test case results, it is possible to conclude that the anti-forensic risk caused by software bugs in digital forensic tools is unacceptable. However, it is important to note the six test cases were deliberately targeted at function areas that were more likely to contain software bugs with associated anti-forensic risk. A more accurate answer to the research question is that software bugs in complex function areas of digital forensic tools pose an unacceptable anti-forensic risk.

Four associated sub-questions were developed to assist in answering the research question. The first sub-question is: What testing approaches are appropriate to test for the presence of software bugs in digital forensic tools? The methodology described in Chapter Three focussed on creating a number of reference sets containing malformed data which were then processed using digital forensic tools. The malformed data was created through a process known as file fuzzing. Figure 4.6 and Figure 4.7 shows the methodology used by the research was successful in identifying ten different software bugs. The answer to the sub-question is that file fuzzing and large reference sets are appropriate for testing for software bugs in digital forensic tools. Importantly, the answer does not mean that other approaches to software testing are ineffective.

The second sub-question is: What test cases and reference sets can be developed to promote and assist the forensic community in testing for and

evaluating software bugs associated with anti-forensic risk? The test cases and reference sets created during the research were successfully used to identify software bugs associated with anti-forensic risks. The answer to the sub-question is that it is relatively simple to create test cases and associated reference sets that could be used by the forensic community for testing. The test cases and reference sets used by the research can serve as examples or templates for use by the forensic community to conduct further testing.

The third sub-question is: How can software bugs in digital forensic tools be ranked and evaluated in terms of severity and anti-forensic risk? Acceptance spectrums used by the research provide increased granularity over a simple pass/fail result that is necessary to give a better indication of the severity of anti-forensic risks. The research analysis presented in Section 4.3 and the discussion of evaluating anti-forensic risk in Section 5.2.2 also demonstrate the importance of investigating and discussing the actual impact of software bugs. The answer to the sub-question is that a spectrum-based ranking system is better than a simple pass/fail system. However no ranking or evaluation methodology can produce accurate results without being coupled with in-depth analysis and discussion.

The fourth sub-question is: What are the risks caused by the presence of software bugs in digital forensic tools? The research analysis presented in Section 4.3 shows that the research identified four distinct types of software bugs with associated anti-forensic risk. Section 5.2.3 discussed the anti-forensic implications of the software bugs in depth and presented a number of possible scenarios where there is a risk of disrupting or preventing an investigation from taking place. The possible existence of more severe risks has been discussed, including hiding of evidence or other more covert methods of subverting an investigation. However, it is important to note that the research has not demonstrated or proven the existence of any severe anti-forensic risks. The answer to the sub-question is therefore that the risks caused by the presence of software bugs in digital forensic tools are that an investigation may be disrupted or prevented.

6.3 LIMITATIONS OF RESEARCH

A number of limitations of the research were first discussed in Section 3.4. The main limitation of the research is that it does not perform widespread testing across all function areas of all digital forensic tools. It was not feasible to conduct testing on a scale that would fully test all digital forensic tools. A subset of function areas from one digital forensic tool likely to contain software bugs was deliberately targeted. The narrow scope of testing performed by the research should be taken into consideration when attempting to generalise the results of the research across all function areas of all digital forensic tools.

A second limitation of the research was the software and hardware in the testing environment. The field testing has been conducted in a single stable environment. However digital forensic tools are used in a wide range of environments. The implication of the use of a single environment is that there may be software bugs identified in the testing environment that cannot be reproduced in other environments. Similarly the testing environment may hide software bugs that are can be observed in other environments.

The initial discussion of limitations in Section 3.4 also discussed the possibility of software bugs that could not be reproduced even within a single stable environment. Software bugs can require complex conditions to trigger them, which can be difficult to reproduce. During the research there was no difficulty encountered in reproducing any of the software bugs observed.

During evaluation and discussion of the anti-forensic implications of the software bugs identified it also became apparent that there are limitations in the ability of the research to evaluate the severity of the associated anti-forensic risks. The field testing and subsequent discussion was successful in identifying a number of software bugs and demonstrating the presence of unacceptable anti-forensic risk. However, the research is not able to conclusively determine the presence or absence of any critically unacceptable anti-forensic risks.

6.4 FURTHER RESEARCH

Recommendations of areas for further research were discussed in depth in Section 5.3. The primary recommendation for further research is to focus on alternative testing methodologies. There is the need to develop alternative methodologies to fill in the gaps in traditional testing methodologies. This research used an alternative methodology composed of techniques such as fuzzing and the use of function mapping and acceptance spectrums. However such alternative testing techniques are still in their infancy and need to be developed and formalised further.

The research successfully used the technique of file fuzzing to successfully identify a number of software bugs. However, the technique of file fuzzing was used in a blind manner to randomly change bytes within a file. File fuzzing can be combined with an understanding of the underlying data structure of a file to intelligently target specific areas of a file with specific changes. Further research could focus on methods to improve the efficiency of file fuzzing as it relates to the testing of digital forensic tools.

The possibility of the software bugs that were identified posing a critically unacceptable anti-forensic risk has been discussed many times throughout the research. Further research could include in depth technical analysis of the software bugs identified to better understand the extent of the associated anti-forensic risk.

Community involvement in testing digital forensic tools could greatly increase the scope and quantity of testing performed. A community-based testing framework could be established to help facilitate widespread testing. Further research could focus on how to structure and standardise test cases, how to distribute test cases and how to authoritatively determine the accuracy of results.

Creation of reference sets to test specific tool function areas takes time to create. The research used fuzzing in order to simplify the creation of reference sets containing large numbers of malformed files. Further research could focus on the creation of tools to automatically create a reference set to test specific functionalities of a tool as selected by the user. The automated reference set created could incorporate targeted fuzzing techniques and also be linked to a community testing framework.

Currently, most testing of digital forensic tools is done in a largely manual manner. The software engineering community has spent decades developing techniques to automate testing of software which can likely be adapted and used by the digital forensic community. However, a major challenge in automating testing of digital forensic tools is that the majority of tools are proprietary black boxes that are not intended to be used in an entirely automated manner. Further research could focus on methods to automate the testing of digital forensic tools.

6.5 CONCLUSION

The research has focused on determining the anti-forensic risks associated with software bugs in digital forensic tools. The literature review in Chapter Two examined a wide range of areas relating to anti-forensic risks. Chapter Three built on the literature reviewed to create a suitable methodology for testing for software bugs and related anti-forensic risk. Field testing was performed based on the methodology with the results presented in Chapter Four and subsequently discussed in depth in Chapter Five. The field testing was successful in identifying a number of software bugs with associated anti-forensic risk. The major finding of the research is that complex function areas of one digital forensic tool contain software bugs that pose an unacceptable anti-forensic risk.

Chapter Six has concluded the thesis by providing a final summary and discussion of the research. A summary of the field findings from Chapter Four was presented in Section 6.1. The research question and sub-questions were answered in Section 6.2. Limitations of the research were discussed in Section 6.3 and areas for further research were summarised in Section 6.4.

The findings presented by the research provide a practical demonstration of the identification of software bugs in digital forensic tools that pose an anti-forensic risk. The research is significant in that it provides tangible evidence of the risks of software bugs in digital forensic tools which provides an incentive for further research. The information provided by the research could be valuable for other researchers interested in anti-forensic risks as well as those interested in alternative testing methodologies for anti-forensic tools.

REFERENCES

Adonis. (2007). *Breaking Encase with FILE0 and Winhex.* Retrieved February 20, 2012, from SafeHack: http://wayback.archive.org/web/jsp/Interstitial.jsp?seconds=5&date=1161459115000&url=http%3A%2F%2Fwww.safehack.com%2FTextware%2Fforensic%2FAnti_Forensic_Break_Encase.pdf&target=http%3A%2F%2Fweb.archive.org%2Fweb%2F20061021193155%2Fhttp%3A%2F%2Fwww.safehac

Ayers, D. (2009, May 1). *Flaws found in "EnCase®" computer forensic software*. Retrived February 1 from NZLawyer Online: http://www.nzlawyermagazine.co.nz/Archives/Issue111/111N2/tabid/1714/Default.aspx

Brenner, S. W., Carrier, B., & Henninger, J. (2005). *The Trojan Horse Defense in Cybercrime Cases.* Retrieved February 25, 2012, from The Center for Education and Research in Information Assurance and Security: www.cerias.purdue.edu/tools_and_resources/bibtex_archive/archive/2005-15.pdf

Brinkmann, M. (2008, July 27). *42 Kilobytes Unzipped Make 4.5 Petabytes*. Retrieved February 5, 2012, from gHacks Technology News: http://www.ghacks.net/2008/07/27/42-kilobytes-unzipped-make-45-petabytes/

Brown, C. L., & Kenneally, E. E. (2005). Risk sensitive digital evidence collection. *Digital Investigation, 2*(2), 101 - 119.

Bunting, S. M. (2008). *Encase Computer Forensics - The Official EnCE: Encase Certified Examiner Study Guide.* Indianapolis: Wiley Publishing, Inc.

Bunting, S. M. (2010, November 13). *Defect in search feature in data view pane (V 6.17 & 6.18 tested)*. Retrieved February 4, 2012, from Guidance

Software Support Portal: https://support.guidancesoftware.com/forum/showthread.php?t=38470

Carrier, B. (2002, October). *Open Source Digital Forensic Tools: The Legal Argument.* Retrieved February 20, 2012, from http://www.cs.plu.edu/courses/netsec/arts/osf.pdf

Carrier, B. (2005). *File System Forensic Analysis* (1st ed.). Upper Saddle River, NJ: Addison-Wesley Professional.

Carrier, B. (2010, August 11). *Digital Forensics Tool Testing Images*. Retrieved February 26, 2012, from Digital Forensics Tool Testing Images: http://dftt.sourceforge.net/

Casey, E. (2011). *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet* (3rd ed.). Waltham, MA: Academic Press.

Charters, I. (2009, 1 January). The Evolution of Digital Forensics: Civilizing the Cyber Frontier. Retrieved 01 11, 2012, from A Perspective on the History of Digital Forensics: http://www.guerilla-ciso.com/wp-content/uploads/2009/01/the-evolution-of-digital-forensics-ian-charters.pdf

Chiang, C. T., Triton, K., & Woodward, A. (2010). An Investigation into the Efficacy of Three Erasure Tools under Windows 7 . *Proceedings of the 8th Australian Digital Forensics Conference.* Edith Cowan University.

Craig, P. (2009, December 17). *RB2: Kiwicon 3 presentation: Hacking Scientists by Paul Craig.* Retrieved February 25, 2012, from Risky Business: http://media.risky.biz/RB2-Paul-Craig-scientists.mp3

Foster, J. C., & Liu, V. T. (2005). Catch Me If You Can. Black Hat USA 2005. Las Vegas.

Garfinkel, S. (2007). Anti-Forensics: Techniques, Detection and Countermeasures. *Proceedings of the 2nd International Conference on Information Warfare & Security* (pp. 77-84). Monterey, California: Academic Conferences Limited.

Garfinkel, S. (2010). Digital forensics research: The next 10 years. *Digital Investigation, 7*(Supplement), S64–S73.

Geiger, M. (2005). Evaluating Commercial Counter-Forensic Tools. *Proceedings of the 5th Annual Digital Forensic Research Workshop,* (pp. 39-41). New Orleans.

Gill, L. (2007, September 7). *Security Advisory: Guidance Software response to iSEC report on EnCase.* Retrieved August 4, 2011, from Computer Security Vulnerabilities: http://securityvulns.com/Rdocument625.html

Grugq, The (2005). *Black Hat USA 2005.* Retrieved February 20, 2012, from http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-grugq.pdf

Guo, Y., & Slay, J. (2010). A Function Oriented Methodology to Validate and Verify Forensic Copy Function of Digital Forensic Tools. *ARES '10 International Conference on Availability, Reliability and Security* (pp. 665 - 670). IEEE Conference Publications.

Guo, Y., Slay, J., & Beckett, J. (2009). Validation and verification of computer forensic software - Seaching function. *Digital Investigation*, S12-22.

Hadnagy, C. (2010). *Social Engineering: The Art of Human Hacking.* Indianiapolis: Wiley Publishing Inc.

Harris, R. (2006). Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. *Digital Investigation, Volume 3*, S44-S49.

Hilley, S. (2007). Anti-forensics with a small army of exploits. *Digital Investigation, 4*, 13-15.

ISO/IEC 17025 Standard. (1999). *General Requirements for the Competence of Testing and Calibration Laboratories.* Geneva: International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC).

Kennedy, D. (2012). *SecManiac*. Retrieved February 3, 2012, from SecManiac: http://www.secmaniac.com/

Klein, T. (2011). *A Bug Hunter's Diary.* San Francisco: No Starch Press.

Leyden, J. (2001, July 23). *DoS risk from Zip of death attacks on AV software?* Retrieved February 5, 2012, from The Register: http://www.theregister.co.uk/2001/07/23/dos_risk_from_zip/

Liu, V. T. (2008). *Metasploit Anti-forensics*. Retrieved February 6, 2012, from The Metasploit Project: http://web.archive.org/web/20080205015610/http://www.metasploit.com/projects/antiforensics/

Liu, V. T., & Stach, P. (2006). Defeating Forensic Analysis. *CEIC 2006.* Las Vegas.

McCullagh , D. (2005, May 24). *Minnesota court takes dim view of encryption*. Retrieved April 3, 2012, from CNET News: http://news.cnet.com/Minnesota%C3%BEcourt%C3%BEtakes%C3%BEdim%C3%BEview%C3%BEof%C3%BEencryption/2100-1030_3-5718978.html

McCash, J. (2010, April 27). *Arbitrary Code Execution on Examiner Systems via File Format Vulnerabilities*. Retrieved February 18, 2012, from SANS Computer Forensic and Incident Response Blog: http://computer-forensics.sans.org/blog/2010/04/27/arbitrary-code-execution-examiner-systems-file-corruption/

McCash, J. (2010a, April 27). *Arbitrary Code Execution on Examiner Systems via File Format Vulnerabilities*. Retrieved February 25, 2012, from Guidance Software Support Portal Forums: https://support.guidancesoftware.com/forum/showthread.php?t=37431

Mills, E. (2008, July 20). *Social Engineering 101: Mitnick and other hackers show how it's done*. Retrieved June 20, 2012, from CNET News: http://news.cnet.com/8301-1009_3-9995253-83.html

National Institute of Standards and Technology (NIST). (2001, November 7). *General Test Methodology for Computer Forensic Tools.* Retrieved February 26, 2012, from Computer Forensic Tool Testing Program: http://www.cftt.nist.gov/Test%20Methodology%207.doc

National Institute of Justice (NIJ). (2008). *Electronic crime scene investigation: a guide for first responders* (2nd ed.). Washington, DC.

Neckar, C. (2010, April 28). *Re: Arbitrary Code Execution on Examiner Systems via File Format Vulnerabilities*. Retrieved Febraury 25, 2012, from Guidance Software Support Portal Forums: https://support.guidancesoftware.com/forum/showpost.php?p=162817&postcount=8

Neckar, C., & Ose, G. (2010, April 23). *Forensic Fail.* Retrieved February 16, 2012, from THOTCON: http://www.thotcon.org/archive/0x1presos/08_THOTCON_0x1-Forensic_Fail-Neckar-Ose.pdf

Newsham, T., Palmer, C., Stamos, A., & Burns, J. (2007, August 1). *Breaking Forensics Software: Weaknesses in Critical Evidence Collection.* Retrieved August 3, 2011, from iSec Partners: http://isecparners.com/storage/white-papers/iSEC-Breaking_Forensics_Software_Paper.v1.1.BH2007.pdf

Oracle. (2012). *Oracle Outside In Technology*. Retrieved February 22, 2012, from Oracle: http://www.oracle.com/us/technologies/embedded/025613.htm

Piper, S., Davis, M., & Shenoi, S. (2006). Countering Hostile Forensic Techniques. *IFIP Advances in Information and Communication Technology, 222/2006,* 79-90.

Richard III, G. G., & Roussev, V. (2006). Next-generation digital forensics. *Communications of the ACM, 49*(2), 76-80.

Ridder, C. K. (2007, August 2). *Evidentiary Implications of Potential Security Weaknesses in Forensic Software.* Retrieved August 4, 2011, from iSec Partners: http://www.isecpartners.com/files/Ridder-Evidentiary_Implications_of_Security_Weaknesses_in_Forensic_Software.pdf

Rogers, M. K. (2005, September 15). *Anti-forensics.* Retrieved January 30, 2012, from Purdue Cyber Forensics: http://cyberforensics.purdue.edu/documents/AntiForensics_LockheedMartin09152005.pdf

Scientific Working Group on Digital Evidence. (2009, January 15). *SWGDE Recommended Guidelines for Validation Testing Version 1.1.* Retrieved February 26, 2012, from Scientific Working Group on Digital Evidence: http://www.swgde.org/documents/current-documents/2009-01-15%20SWGDE%20Recommendations%20for%20Validation%20Testing%20Version%20v1.1.pdf

Slay, J., & Beckett, J. (2007). Digital Forensics: Validation and Verification in a Dynamic Work Environment. *40th Annual Hawaii International Conference on System Sciences* (pp. 266a-276a). IEEE Conference Publications.

Sommer, P. (2010). Forensic science standards in fast-changing environments. *Science & Justice, 50*(1), 12-17.

Stamm, M. C., Tjoa, S. K., Lin, S. W., & Liu, R. K. (2010). Anti-forensics of JPEG compression. *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (pp. 1694 - 1697). IEEE Conference Publications.

Stewart, J. (2010, April 27). *Arbitrary Code Execution on Examiner Systems via File Format Vulnerabilities*. Retrieved from SANS Computer Forensic and Incident Response Blog: http://computer-forensics.sans.org/blog/2010/04/27/arbitrary-code-execution-examiner-systems-file-corruption/

Stockdale, A. (2010, April 27). *Re: Arbitrary Code Execution on Examiner Systems via File Format Vulnerabilities*. Retrieved February 25, 2012, from Guidance Software Support Portal Forums: https://support.guidancesoftware.com/forum/showpost.php?p=162772&postcount=3

Sutton, M., Greene, A., & Amini, P. (2007). *Fuzzing: Brute Force Vulnerability Discovery*. Upper Saddle River, NJ: Addison-Wesley.

Verisign iDefense Security. (2009, May 15). *Multiple Vendor Outside In Multiple Spreadsheet Buffer Overflow Vulnerabilities*. Retrieved February 23, 2012, from Verisign iDefense Security: http://www.verisigninc.com/en_US/products-and-services/network-intelligence-availability/idefense/public-vulnerability-reports/articles/index.xhtml?loc=en_US&id=801

Whitteker, M. (2008, November). Breaking the Forensic Process. *ISSA Journal*, 10-16.

Wilsdon, T., & Slay, J. (2006). Validation of Forensic Computing Software Utilizing Black Box Testing Techniques. *Proceedings of the 4th Australian Digital Forensics Conference.* Perth, Western Australia: Edith Cowan University.

Appendix A – Reference Sets

Reference Set ID:

RS-IMAGE-01

Description:

A set of malformed images generated by fuzzing the images from the GovDocs1 dataset (http://digitalcorpora.org/corpora/files).

Original Files:

File	MD5	Source
020877.bmp	849a4918b8303bcc1db9f0ce0de23fe4	GovDocs1
020879.bmp	733e3cd6373535c076e4c10aa4012ef0	GovDocs1
020881.gif	4d74c4c46fda204997fa2f27127bcf33	GovDocs1
020882.gif	4affcc6fa64ce714d7423bffb8e9333b	GovDocs1
020264.jpg	5be169056d2124490fd8eadc0ffa49ea	GovDocs1
020878.jpg	a9acd7740c26cff6f22f2f99fcbf1ae3	GovDocs1
020714.png	2be5473c03735cb47b226f564eb835e9	GovDocs1
020718.png	bceb965d15d30d68fb33b25a02197986	GovDocs1

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
020877.bmp	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020879.bmp	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020881.gif	0.1	2000
	0.2	2000
	0.5	2000

Original File	Fuzzing Percentage	Number Of Files
	1	2000
	2	2000
020882.gif	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020264.jpg	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020878.jpg	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020714.png	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000
020718.png	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000

Reference Set ID:

RS-EMAIL-01

Description:

A set of malformed PST files created by fuzzing. The original PST file contains 20 emails from the Enron dataset and was created using Nuix 3.6.7.

Original Files:

File	MD5	Source
Export.pst	c03a22a6a75a03cbcdce42ca0509c4ea	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Export.pst	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000

Reference Set ID:

RS-EMAIL-02

Description:

A set of malformed NSF files created by fuzzing. The original NSF file contains 20 emails from the Enron dataset and was created using Nuix 3.6.7.

Original Files:

File	MD5	Source
Export.nsf	7fb90dd4d05e781c81ceeafe84e4f1b0	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Export.nsf	0.1	500
	0.2	500
	0.5	500
	1	500
	2	500

Reference Set ID:

RS-EMAIL-03

Description:

A set of malformed MBOX files created by fuzzing. The original MBOX file contains 20 emails from the Enron dataset and was created using Nuix 3.6.7.

Original Files:

File	MD5	Source
Export.mbox	b58564cc242522afdb49709681b1f5f1	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Export.mbox	0.1	2000
	0.2	2000
	0.5	2000
	1	2000
	2	2000

Reference Set ID:

RS-EMAIL-04

Description:

A set of malformed DBX files created by fuzzing. The original file is from the "Hacking Case" reference image provided by CFReDS (http://www.cfreds.nist.gov/Hacking_Case.html) and is located in the "C:\Documents and Settings\Mr. Evil\Local Settings\Application Data\Identities\{EF086998-1115-4ECD-9B13-9ADC067B4929}\Microsoft\Outlook Express\" directory.

Original Files:

File	MD5	Source
Inbox.dbx	69b8fbb96823671d01c3aba467647114	CFReDS Hacking Case

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Inbox.dbx	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-CONTAINER-01

Description:

A set of malformed ZIP files created by fuzzing. The original file is from "Container Files" reference image provided by CFReDS (http://www.cfreds.nist.gov/SearchingContainerFiles.html).

Original Files:

File	MD5	Source
archive-zip.zip	43aad4c41c53f654ad30095144485669	CFReDS Container Files

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
archive-zip.zip	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-CONTAINER-02

Description:

A set of malformed GZIP files created by fuzzing. The original file is from "Container Files" reference image provided by CFReDS (http://www.cfreds.nist.gov/SearchingContainerFiles.html).

Original Files:

File	MD5	Source
archive-tar_gzip.tar.gz	f764c4ad8a6857385f5d8e20f065be50	CFReDS Container Files

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
archive-tar_gzip.tar.gz	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-CONTAINER-03

Description:

A set of malformed TAR files created by fuzzing. The original file is from "Container Files" reference image provided by CFReDS (http://www.cfreds.nist.gov/SearchingContainerFiles.html).

Original Files:

File	MD5	Source
archive-tar.tar	39de76427db926fbfb2ff1333283831b	CFReDS Container Files

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
archive-tar.tar	0.1	1000
	0.2	1000
	0.5	1000
	1	1000
	2	1000

Reference Set ID:

RS-CONTAINER-04

Description:

A set of malformed RAR files created by fuzzing. The original file is from "Container Files" reference image provided by CFReDS (http://www.cfreds.nist.gov/SearchingContainerFiles.html).

Original Files:

File	MD5	Source
archive-rar.rar	acd6267de4960a0a1534da859476d2f8	CFReDS Container Files

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
archive-rar.rar	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-CONTAINER-05

Description:

A set of malformed BZIP2 files created by fuzzing. The original file is from "Container Files" reference image provided by CFReDS (http://www.cfreds.nist.gov/SearchingContainerFiles.html).

Original Files:

File	MD5	Source
archive-tar_bzip2.tar.bz2	9ee38808d2065f9fa997e63392f49f6b	CFReDS Container Files

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
archive-tar_bzip2.tar.bz2	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-INTERNET-01

Description:

A set of malformed Firefox history/bookmark files. The original file is from the nps-2009-domexusers disk image (available at http://digitalcorpora.org/corpora/disk-images) and is located in the "C:\Documents and Settings\domex2\Application Data\Mozilla\Firefox\Profiles\n2utfxqg.default\" directory.

Original Files:

File	MD5	Source
places.sqlite	045e8c52126d673ab5405294e9173e96	nps-2009-domexusers

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
places.sqlite	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-INTERNET-02

Description:

A set of malformed Internet Explorer history files. The original file is from the "Hacking Case" reference image provided by CFReDS (http://www.cfreds.nist.gov/Hacking_Case.html) and is located in the "C:\Documents and Settings\Mr. Evil\Local Settings\History\History.IE5\MSHist012004081620040823\" directory.

Original Files:

File	MD5	Source
index.dat	d022289cd71993a723744c3683be7ba1	CFReDS Hacking Case

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
index.dat	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-INTERNET-03

Description:

A set of malformed Opera history files. The original file was created by installing Opera and then visiting a number of websites.

Original Files:

File	MD5	Source
global_history.dat	d32909ad79c71b73779a1572686644e5	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
global_history.dat	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-INTERNET-04

Description:

A set of malformed Safari history files. The original file was created by installing Safari and then visiting a number of websites.

Original Files:

File	MD5	Source
History.plist	cc15fdaa420831476a01669a81f3a2e4	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
History.plist	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-INTERNET-05

Description: A set of malformed Chrome history files. The original file was created by installing Chrome and then visiting a number of websites.

Original Files:

File	MD5	Source
History	96d1bdb7e2bd42a3f6cd3e458271b6b5	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
History	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-WINDOWS-01

Description:

A set of malformed Windows link files. The original file is from the "Hacking Case" reference image provided by CFReDS (http://www.cfreds.nist.gov/Hacking_Case.html) and is located in the "C:\Documents and Settings\All Users\Start Menu\Programs\" directory.

Original Files:

File	MD5	Source
Windows Messenger.lnk	883fcd088fb230b9811ccacb96b7d1f7	CFReDS Hacking Case

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Windows Messenger.lnk	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-WINDOWS-02

Description:

A set of malformed Windows INFO2 files. The original file is from the "Hacking Case" reference image provided by CFReDS (http://www.cfreds.nist.gov/Hacking_Case.html) and is located in the "C:\RECYCLER\S-1-5-21-2000478354-688789844-1708537768-1003\INFO2" directory.

Original Files:

File	MD5	Source
INFO2	feb4b138086193c9efd98877c93b1a2d	CFReDS Hacking Case

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
INFO2	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-LOG-01

Description:

A set of malformed Windows legacy event log files. The original file is from the "Hacking Case" reference image provided by CFReDS (http://www.cfreds.nist.gov/Hacking_Case.html) and is located in the "C:\WINDOWS\system32\config\" directory.

Original Files:

File	MD5	Source
SysEvent.Evt	99c704081bf7fc942695adb80e983f01	CFReDS Hacking Case

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
SysEvent.Evt	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Reference Set ID:

RS-LOG-02

Description:

A set of malformed Windows event log files. This event log was taken from the test computer which was running Windows 7 Enterprise (64-bit).

Original Files:

File	MD5	Source
Microsoft-Windows-NetworkProfile-Operational.evtx	6ed2c8bc7705d7bbb15dc7bcfbbe4369	Created

Reference Set Contents:

Original File	Fuzzing Percentage	Number Of Files
Microsoft-Windows-NetworkProfile-Operational.evtx	0.1	5000
	0.2	5000
	0.5	5000
	1	5000
	2	5000

Appendix B – Test Cases

Test Case ID:

TC.01

Requirements Tested:

EC.EP.01

Reference Sets Used:

RS-IMAGE-01 (JPG, PNG, BMP & GIF)

Methodology:

1. Each reference set is individually loaded onto a blank hard drive
2. A new EnCase case file is created
3. The hard drive containing the reference set is added to the case
4. Evidence Processor is started with only the "Thumbnail Creation" function selected

Results:

- EnCase successfully processed RS-IMAGE-01 without showing any indications of software bugs.

Acceptance Spectrum Determination:

Pass – Exceeds Expectations

Test Case ID:

TC.02

Requirements Tested:

EC.EP.02

Reference Sets Used:

RS-EMAIL-01 (PST)

RS-EMAIL-02 (NSF)

RS-EMAIL-03 (MBOX)

RS-EMAIL-04 (DBX)

Methodology:

1. Each reference set is individually loaded onto a blank hard drive
2. A new EnCase case file is created
3. The hard drive containing the reference set is added to the case

4. Evidence Processor is started with only the “Find Email” function selected and only the specific email format used by the reference set selected (e.g. PST). The “Search for Additional Lost or Deleted Items” option is also selected.

Results:

- RS-EMAIL-01 caused EnCase to crash with an error message within minutes of starting processing; this result is repeatable.
- RS-EMAIL-02 caused EnCase to crash with an error message within minutes of starting processing; this result is repeatable.
- EnCase successfully processed RS-EMAIL-03 without showing any indication of software bugs.
- Processing of RS-EMAIL-04 was abandoned after 24 hours at approximately 75% completion as EnCase had filled a 2TB drive with logical evidence files in the case cache folder. (3GB of DBX files expanding to 1.8TB).

Acceptance Spectrum Determination:

Fail – Unacceptable

Further research needed to determine if bugs pose a critical anti-forensic risk.

Test Case ID:

TC.03

Requirements Tested:

EC.EP.03

Reference Sets Used:

RS-CONTAINER-01 (ZIP)

RS-CONTAINER-02 (GZIP)

RS-CONTAINER-03 (TAR)

RS-CONTAINER-04 (RAR)

RS-CONTAINER-05 (BZIP2)

Methodology:

1. Each reference set is individually loaded onto a blank hard drive
2. A new EnCase case file is created
3. The hard drive containing the reference set is added to the case

4. Evidence Processor is started with only the “Expand Compound Files” function selected.

Results:

- RS-CONTAINER-01 caused EnCase to exit unexpectedly without any error messages during the “Processing Artifacts” stage after approximately 18 hours of processing. This result has not been tested to see if it is repeatable.
- RS-CONTAINER-02 caused EnCase to crash with an error message at the end of “Processing Artifacts” stage after approximately 2 days of processing. This result has not been tested to see if it is repeatable.
- EnCase successfully processed RS-CONTAINER-03 without showing any indication of software bugs.
- RS-CONTAINER-04 caused EnCase to crash with an error message within minutes of the starting of processing; this result is repeatable.
- RS-CONTAINER-05 caused EnCase to crash with an error message within minutes of the starting of processing; this result is repeatable.

Acceptance Spectrum Determination:

Fail – Unacceptable

Further research needed to determine if bugs pose a critical anti-forensic risk.

Test Case ID:

TC.04

Requirements Tested:

EC.EP.04

Reference Sets Used:

RS-INTERNET-01 (Firefox)

RS-INTERNET-02 (Internet Explorer)

RS-INTERNET-03 (Opera)

RS-INTERNET-04 (Safari)

RS-INTERNET-05 (Chrome)

Methodology:

- Each reference set is individually loaded onto a blank hard drive
- A new EnCase case file is created

- The hard drive containing the reference set is added to the case
- Evidence Processor is started with only the “Find internet artifacts” function selected. The “Search unallocated space for internet artifacts” option is unselected.

Results:

- RS-INTERNET-01 caused EnCase to crash with an error message within minutes of the starting of processing; this result is repeatable.
- EnCase successfully processed RS-INTERNET-02 without showing any indication of software bugs.
- EnCase successfully processed RS-INTERNET-03 without showing any indication of software bugs. The processing was very fast and no results were returned which may mean EnCase could not recognise any files as valid internet artifacts.
- RS-INTERNET-04 caused EnCase to crash with an error message within minutes of the starting of processing; this result is repeatable.
- EnCase successfully processed RS-INTERNET-05 without showing any indication of software bugs.

Acceptance Spectrum Determination:

Fail – Unacceptable

Further research needed to determine if bugs pose a critical anti-forensic risk.

Test Case ID:

TC.05

Requirements Tested:

EC.EP.05

Reference Sets Used:

RS-WINDOWS-01 (LNK)

RS-WINDOWS-02 (INFO2)

Methodology:

- Each reference set is individually loaded onto a blank hard drive
- A new EnCase case file is created
- The hard drive containing the reference set is added to the case

- Evidence Processor is started with only the “Windows Artifact Parser” function selected and only the option for the specific artifact being searched for selected (e.g. LNK files). The “Search Unallocated” option is not selected.

Results:

- EnCase successfully processed RS-WINDOWS-01 without showing any indication of software bugs.
- EnCase successfully processed RS-WINDOWS-02 without showing any indication of software bugs.

Acceptance Spectrum Determination:

Pass – Exceeds Expectations

Test Case ID:

TC.06

Requirements Tested:

EC.EP.06

Reference Sets Used:

RS-LOG-01 (EVT)

RS-LOG-02 (EVTX)

Methodology:

- Each reference set is individually loaded onto a blank hard drive
- A new EnCase case file is created
- The hard drive containing the reference set is added to the case
- Evidence Processor is started with only the “Windows Event Log Parser” function selected. No event conditions are set.

Results:

- EnCase successfully processed RS-LOG-01 without showing any indications of software bugs.
- EnCase generates an “internal error” within a minute of the starting of processing RS-LOG-02. This result is repeatable.

Acceptance Spectrum Determination:

Pass – Meets expectations

The software bug detected did not result in EnCase crashing.

Appendix C – Test Journal

Date	Action	Result/Notes
10/05/2012	Created file fuzzer in C# and tested fuzzing	Fuzzer worked as expected
24/05/2012	Built PC for testing	Bios needed to be flashed to F8 version to support CPU model
26/05/2012	Started function mapping of Encase 7.04 Evidence Processor	Function mapping should focus on mapping "inputs" instead of trying to fully map all functions
	Created sample set of 10,000 fuzzed jpeg images at 0.1% fuzzing	Fuzzer took less than a minute to create sample set (approximately 9 GB in size)
	Input jpeg sample set into Evidence Processor and selected thumbnail generation only	Encase took approximately 4 minutes to process the sample set
	Created sample set of 10,000 fuzzed zip archives at 0.1% fuzzing	Fuzzer took approximately 1 minute to create sample set (approximately 16 GB in size)
	Input zip sample set into Evidence Processor and selected expand compound files only	First attempt failed; note to ensure that evidence cache is located on volume with sufficient free space. Second attempt successful; Encase took approximately 140 minutes to process the sample set, 9907/10000 files made it to the "processing Artifiacts" stage.
3/06/2012	Continued function mapping of Encase 7.04 Evidence Processor	Mostly successful, a number of areas are difficult to map entirely
	Started creating test requirements and test cases	
	Created reference Set RS-IMAGE-01	References sets should be able to be distributed as the original files + a script to generate a fuzzed set Interestingly Windows Security Essentials detected 4 of the fuzzed files as containing a JPEG exploit
	Started running test case TC.01	
6/06/2012	Finished running test case TC.01	Pass result - took 2 days 12 hours to process the set

Date	Action	Result/Notes
8/06/2012	Created reference sets RS-EMAIL-01 to RS-EMAIL-03	
	Started and finished running test case TC.02 - reference Set RS-EMAIL-01	EnCase crashed after approximately 1 minute of processing. Repeated test and got the same result.
	Started and finished running test case TC.02 - reference Set RS-EMAIL-02	EnCase crashed after approximately 1 minute of processing. Repeated test and got the same result.
	Started and finished running test case TC.02 - reference Set RS-EMAIL-03	Pass result - took 2 hours 40 minutes to process the set. Note that EnCase does not recognise items if the file signature portion of the file has been fuzzed.
12/06/2012	Created reference set RS-EMAIL-04	
	Started running test case TC.02 - reference Set RS-EMAIL-04	Filled up SSD that was being used for cache (120 GB), restarted with 2TB drive as cache. Possibly some issue resulting in very large L01s from malformed DBX files
	Created reference sets RS-CONTAINER-01 to RS-CONTAINER-05	
13/06/2012	Finished running test case TC.02 - reference Set RS-EMAIL-04	Abandoned after 24 hours, filled up 2TB drive with logical evidence files in cache. Possible issue with processing DBX files that leads to incorrectly sized extracts.
	Started running test case TC.03 - reference Set RS-CONTAINER-01	
14/06/2012	Finished running test case TC.03 - reference Set RS-CONTAINER-01	EnCase exited unexpectedly (no error messages) in the "Processing Artifacts" stage after approximately 18 hours
	Started running test case TC.03 - reference Set RS-CONTAINER-02	
	Created reference sets RS-INTERNET-01 to RS-INTERNET-05	

Date	Action	Result/Notes
16/06/2012	Finished running test case TC.03 - reference Set RS-CONTAINER-02	EnCase crashed at the end of the "Processing Artifiacts" stage after 2 days of processing. Possibly related to issue seen with RS-CONTAINER-01; however there was a definite crash with an error message this time. Computer froze up when attempting to close the error message and exit EnCase.
18/06/2012	Started and finished running test case TC.03 - reference Set RS-CONTAINER-03	EnCase processed the reference set without any issues
	Started and finished running test case TC.03 - reference Set RS-CONTAINER-04	EnCase crashed after approximately 1 minute of processing.
	Started and finished running test case TC.03 - reference Set RS-CONTAINER-05	EnCase crashed after approximately 1 minute of processing.
	Started and finished running test case TC.04 - reference Set RS-INTERNET-01	EnCase crashed after approximately 1 minute of processing.
	Started and finished running test case TC.04 - reference Set RS-INTERNET-02	EnCase processed the reference set without any issues
	Started and finished running test case TC.04 - reference Set RS-INTERNET-03	EnCase processed the reference set without any issues. Note that processing was very quick and no results were returned. Possibly something in the data structure that indicates if the file is corrupt.
	Started and finished running test case TC.04 - reference Set RS-INTERNET-04	EnCase crashed after approximately 1 minute of processing.
	Started and finished running test case TC.04 - reference Set RS-INTERNET-05	EnCase processed the reference set without any issues
23/06/2012	Started and finished running test case TC.05 - reference Set RS-WINDOWS-01	EnCase processed the reference set without any issues
	Started and finished running test case TC.05 - reference Set RS-WINDOWS-02	EnCase processed the reference set without any issues
	Started running test case TC.06 - reference Set RS-LOG-01	

Date	Action	Result/Notes
26/06/2012	Finished running test case TC.06 - reference Set RS-LOG-01	EnCase processed the reference set without any issues
	Started and finished running test case TC.06 - reference Set RS-LOG-02	EnCase generates an "internal error" soon after starting processing which results in the Case Processor stopping. This error prevents the Case Processor from being restarted.
28/06/2012	Reran test case TC.06 - reference Set RS-LOG-02	Results are repeatable with 42 seconds from the starting of processing to error.

Appendix D – File Fuzzer Source Code

The C# source code for the file fuzzer used to create the reference sets is presented below. The source code was compiled using Visual Studio 2010 with a target framework of .NET Framework 4 Client Profile.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Security.Cryptography;

namespace Fuzzer
{
    class Program
    {
        static void Main(string[] args)
        {
            /* Arguments:
             * 0 - Input File
             * 1 - Output Folder
             * 2 - Number of files to generate
             * 3 - Fuzzing percentage
             * 4 - Fuzzing Seed (optional)
             */

            string inputFile = null;
            string outputFolder = null;
            int numberToGenerate = 0;
            double fuzzingPercentage = 0;
            int? fuzzingSeed = null;

            if (args.Count() >= 4)
            {
                inputFile = args[0];
                outputFolder = args[1];
                numberToGenerate = Convert.ToInt32(args[2]);
                fuzzingPercentage = Convert.ToDouble(args[3]);
                if (args.Count() == 5)
                {
                    fuzzingSeed = Convert.ToInt32(args[4]);
                }
            }
            else
            {
                tsWriteLine("ERROR: Invalid Arguements");
                Environment.Exit(-1);
            }

            byte[] inputFileBytes = null;


            // Read input file
            try
            {
                tsWriteLine("Opening " + inputFile);
                inputFileBytes = File.ReadAllBytes(inputFile);
                tsWriteLine("Total Bytes: " + inputFileBytes.Length);
```

```
                tsWriteLine("First Byte: " +
inputFileBytes[0].ToString("X2"));
                tsWriteLine("Last Byte: " +
inputFileBytes[inputFileBytes.Length - 1].ToString("X2"));
            }
            catch (System.IO.DirectoryNotFoundException ex)
            {
                tsWriteLine("ERROR: Directory Not Found");
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }
            catch (System.IO.FileNotFoundException ex)
            {
                tsWriteLine("ERROR: File Not Found");
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }
            catch (System.UnauthorizedAccessException ex)
            {
                tsWriteLine("ERROR: File Access Exception");
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }
            catch (System.IO.IOException ex)
            {
                tsWriteLine("ERROR: File IO Exception");
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }
            catch (Exception ex)
            {
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }


            // Check if output path exists, if not create it
            try
            {
                if (!Directory.Exists(outputFolder))
                {
                    Directory.CreateDirectory(outputFolder);
                    tsWriteLine("Creating folder: " + outputFolder);
                }
            }
            catch (Exception ex)
            {
                tsWriteLine(ex.Message);
                Environment.Exit(-1);
            }
            tsWriteLine("Output Folder: " + outputFolder);

            // Get output file number padding length
            string outputFilePaddingLength = "D" +
numberToGenerate.ToString().Length.ToString();
            // Get input file extension
            string inputFileExtension = Path.GetExtension(inputFile);
            // Get number of bytes to fuzz
            int numberBytesToFuzz = (int)((fuzzingPercentage / 100) *
inputFileBytes.Length);
            tsWriteLine(numberBytesToFuzz.ToString() + " bytes (" +
fuzzingPercentage.ToString() + "%) will be fuzzed");
```

```
            tsWriteLine("Generating files...");

            // Generate fuzzed files
            int updateCounter = 0;
            for (int i = 0; i < numberToGenerate; i++)
            {
                int randomSeed = 0;
                // Check for manually specified fuzzing seed
                if (fuzzingSeed == null)
                {
                    // Generate a new random seed
                    RNGCryptoServiceProvider seedGenerator = new
RNGCryptoServiceProvider();
                    byte[] randomBytes = new byte[4];
                    seedGenerator.GetBytes(randomBytes);
                    randomSeed = BitConverter.ToInt32(randomBytes, 0);
                }
                else
                {
                    // Use the specified seed
                    randomSeed = fuzzingSeed.Value;
                }

                // Set the output file path
                string outputPath = outputFolder + @"\" + (i +
1).ToString(outputFilePaddingLength) + " (" + randomSeed + ")" +
inputFileExtension;

                // Output a fuzzed file
                WriteFuzzedFile(inputFileBytes, outputPath,
numberBytesToFuzz, randomSeed);

                // Output a status update every 100 files
                updateCounter++;
                if (updateCounter == 100)
                {
                    tsWriteLine((i + 1) + " of " + numberToGenerate + "
generated...");
                    updateCounter = 0;
                }
            }

            tsWriteLine("Finished");
        }

        // Prefixes the output line with a timestamp
        static void tsWriteLine(string text)
        {
            Console.WriteLine("[" + DateTime.Now.ToString("HH:mm:ss") + "]
" + text);
        }

        static void WriteFuzzedFile(byte[] inputFile, string outputFile,
int bytesToFuzz, int randomSeed)
        {
            // Create a clone of the existing byte array so the original
data is not modified
            byte[] fuzzByteArray = (byte[])inputFile.Clone();

            // Set up random number generator for fuzzing bytes
            Random rng = new Random(randomSeed);
```

```
            for (int i = 0; i < bytesToFuzz; i++)
            {
                // Randomly select a byte from the file
                int byteToFuzz = rng.Next(0, fuzzByteArray.Length);
                // Generate a random byte
                byte[] randomByte = new byte[1];
                rng.NextBytes(randomByte);
                // Replace the byte in the file
                fuzzByteArray[byteToFuzz] = randomByte[0];
            }

            // Output the file
            try
            {
                File.WriteAllBytes(outputFile, fuzzByteArray);
            }
            catch (Exception ex)
            {
                tsWriteLine(ex.Message);
            }
        }
    }
}
```