A Novel Forensic Triage Approach for the Windows Operating System

Charley Célice - 10012794

Submitted in partial fulfilment of the requirements of Edinburgh Napier University for the Degree of BEng (Hons) Computer Security and Forensics

In collaboration with SCDEA

School of Computing

April 2014
Authorship Declaration

I, Charley Célice, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed:

Date:

Matriculation no: 10012794
Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.
Abstract

Digital Forensics has been an exponentially growing area where a good number of crimes are committed from a range of electronic devices such as computers, mobile phones and even tablets. A key factor in the undergoing of a digital forensic investigation is the increasing amount of data to be processed to find digital evidence, notably when dealing with computers. This has created scenarios where investigations could take up to several days before any type of evidence is retrieved.

This dissertation looks at ways on how computer triage is used and whether there is a better approach to use considering multiple factors such as the amount of information retrieved in a short amount of time, and research on current methodologies and scenarios. Multiple triage applications were looked into, as well as collaboration with the Scottish Crime and Drug Enforcement Agency (SCDEA), in order to propose a technique which can be used in the development of a new type of computer triage tool.

Amongst the three most available Operating Systems to the general public, the Windows platform was chosen for experimentation for its popularity. The tool was then developed in Python and assessed using the appropriate metrics based on researched triage assessment methods and current tools such as COFEE and TriageIR. For the evaluation, the prototype tool was run on a range of experiments and the collected results compared. Its functionality and performance was then also compared to another open-source triage tool: TriageIR.

After successfully implementing and evaluating the prototype tool, not without issues, and comparing its functionalities with another open-source triage tool, it was concluded that an approach in quickly determining a course of action after running a simple application could help make digital forensic processes generally quicker. The investigator is now able to know as soon as possible if any traces of suspicious activity are present on the machine, and do not have to wait for a tool to perform complete analysis of the entire media. From that, they can safely assume that there is a high possibility for interesting evidence being present on the digital device.
List of Tables

Table 1 - W3School's OS Platform Trends for 2013 (W3SCHOOL, 2013) ........ 25
Table 2 - Experimental Environment Setup .................................................. 43
Table 3 - Experiment Results Tables (IIOC) .................................................... 58
Table 4 - Experiment Results (Fraud) ............................................................ 61
List of Figures

Figure 1 - Evidence Reliability ........................................................................................................... 21
Figure 2 - NetMarketShare’s OS Platform Trends for 2013 (NetMarketShare, 2013) ......................... 26
Figure 3 - Overview Experimental Design .......................................................................................... 34
Figure 4 - Files by Hash Value Design Diagram ................................................................................. 37
Figure 5 - Files by Hash Value Pseudo code......................................................................................... 37
Figure 6 - Encryption Detection Design Diagram ................................................................................ 38
Figure 7 - Encryption Detection Pseudo Code ..................................................................................... 39
Figure 8 - Text Based Searching Design Diagram .............................................................................. 40
Figure 9 - Text Based Searching Pseudo Code .................................................................................... 40
Figure 10 - Main Window Folder Structure ......................................................................................... 44
Figure 11 - Main Window Code part 1 .................................................................................................. 45
Figure 12 - Main Window Code part 2 .................................................................................................. 46
Figure 13 - Module Loader Code ......................................................................................................... 46
Figure 14 - Module Loader Configuration File .................................................................................... 47
Figure 15 - Main Screen Layout Starting .............................................................................................. 47
Figure 16 - Main Screen Layout Drive Selection ................................................................................... 47
Figure 17 - Main Screen Layout Confirmation ..................................................................................... 48
Figure 18 - Main Screen Layout Modules Call ..................................................................................... 48
Figure 19 - Files by Hash Value Folder Structure ................................................................................ 48
Figure 20 - Files by Hash Value Code part 1 ....................................................................................... 49
Figure 21 - Files by Hash Value Configuration File .............................................................................. 49
Figure 22 - Files by Hash Value part 2 ................................................................................................ 50
Figure 23 - Files by Hash Value Generated Alerts ............................................................................... 50
Figure 24 - Encryption Detection Folder Structure .............................................................................. 51
Figure 25 - Encryption Detection Code part 1 ...................................................................................... 51
Figure 26 - Encryption Detection Code part 2 ..................................................................................... 51
Figure 27 - Encryption Detection Configuration File ............................................................................. 52
Figure 28 - Encryption Detection Generated Alerts ............................................................................. 52
Figure 29 - Text-Based Searching Folder Structure ............................................................................. 52
Figure 30 - Text-Based Searching Configuration File .......................................................................... 53
Figure 31 - Text-Based Searching Code part 1 .................................................................................... 53
Figure 32 - Text-Based Searching Code part 2 .................................................................................... 54
Figure 33 - Text-Based Searching Generated Alerts ............................................................................ 54
Figure 34 - Total Execution Time and Time of First Discovered Evidence ....................................... 59
Acknowledgements

I would like to thank first Richard Macfarlane and Prof. Bill Buchanan for their amazing support and knowledge, as well as providing access to the cloud infrastructure used during the project.

I would also like to thank all the students and researchers of Edinburgh Napier University for their collaboration and sharing of advice.

In addition, thanks to Mike Dickson from the Scottish Crime and Drug Enforcement Agency (SCDEA). His input was a really important source of help for this project.

Finally, thanks to my parents and my partner for the support they have been giving me all those past years.
1 Introduction

1.1 Context

Nowadays, an important number of crimes are performed from an increasing number of different digital devices (S. L. Garfinkel, 2010). This has created scenarios where on-scene investigations can include many different devices which analysis has to be performed on (Dickson, 2013); a good example of this is when multiple machines have to be seized in a corporate environment, whether or not it is known that they contain digital evidence at all. Moreover, the impact of the exponential growing of memory in electric devices also cannot be underestimated; with growth discussed often by researchers: “6.65 times (from 84 GB to 559 GB) in eight years (2003-2011)” (Roussev, Quates, & Martell, 2013). This is an important factor in digital forensics as it requires techniques to evolve in order to perform useful forensic analysis in short periods of time. Thus, in order to overcome the problems of increasing number of devices and the constraints of memory growth of electronic devices in digital forensics, solutions such as forensic triage techniques have to be developed.

1.2 Background

1.2.1 Computer Forensics

Digital Forensics is composed of different methods, processes, and approaches such as Evidence Collection and Analysis. This particular type of investigation was mostly done by government agencies but is now often also common in the commercial sector (Carrier, 2002). Companies have the need of dealing efficiently with Incident Response, meaning to deal with an incident once it has already happened. This involves several challenges to overcome, which can be regrouped into different Digital Forensics methods.

Computer Forensics can be defined as collecting, preserving, analysing and presenting electronic evidence to solve a legal matter (Daniel & Daniel, 2011). Each of those steps uses different techniques and all follow very strict procedures that are looked into more deeply later in the chapter, but can be summarised as:

- Collection – Typically the gathering of information on a device.
- Preservation – Keeping the gathered information safe.
- Analysis – Examine the data to find traces of evidence.
- Presentation – Present the evidence, usually to a court audience.
1.2.2 Forensic Tools and Approaches

The forensic tools used should be “reliable and relevant” (Carrier, 2002). This means that not any tool can be used for gathering evidence, as it first has to be tested and approved, for example peer-reviewed, or supported by an important piece of research so they are seen trustful to the eyes of law enforcement.

In Digital Forensics, typically various techniques are used in different situations. Different approaches exist, right from the collection phase. For this step, the most popular way is typically to directly collect all information on a seized machine, for example either directly from an off powered computer, or by doing a live forensic investigation on-scene. The second way would be to first gather, and in a more rapid way, information on the device which may help the investigator find the most relevant artefacts, thus saving time; this is known as Triaging (Roussev & Quates, 2012).

These techniques are not only helping to solve criminal cases by the law enforcement, but can also be used by companies such as for Incident Response, which in 2013 has been recognised as a significant problem to overcome even in the eyes of the information security arm of the Government Communications Headquarters (GCHQ) and the Council of Registered Ethical Security Testers (CREST) amongst others, who just recently “launched two cyber incident response schemes” to help targeted businesses (Leyden, 2013).

1.3 Aims and Objectives

The principal aim of this project is to perform a thorough literature review of the various techniques used in today’s digital forensics procedures and the current needs in order to consider a best approach, and design and implement and evaluate a forensically sound and useful new application. This application should perform forensic analysis on an electronic device considering today’s Digital Forensic challenges and already existing tools, based on the literature and technical reviews carried out. In order to achieve this, these objectives must be met:

- Critically review the different approaches and techniques used in Digital Forensics, including the current different available tools, as well as defining the importance of the different parameters in a given scenario, and which methods are becoming important in today’s investigations. It will also be important to understand the legal aspects involved with such methods, for example in a corporate environment.
- Design an application based on the research and the review of other tools, and design a range of experiments for the evaluation, such as various real world based scenarios.
- Implement the application and run it against the different experiments.
• Evaluate the results of the implementation of the final prototype, and the associated experiments, such as the amount of useful analysis, evidence gathered, time span on different scenarios, generated alerts, the amount of resources used and so on.

• If possible, start researching into a better way to store and display the acquired information and evidence such as using comprehensible charts and graphs, or an XML format standard to Digital Forensics, which could lead towards a deeper research on data visualisation in Digital Forensics.

1.4 Dissertation Structure

This dissertation is divided into nine main chapters (excluding the introduction), as follows:

Chapter 2 provides a literature review, critically analysing the different techniques and approaches of Digital Forensics, after having explained the basic methodology. From there, feedback from SCDEA is provided to have a better overview of today’s real-world requirements. Following this, methods such as Triage and traditional Digital Forensics are compared. Finally, research into the tools developed so far in regards to such methods is performed, as well as a look at the legal aspects including the Association of Chief Police Officers (ACPO) guidelines for electronic evidence. Garfinkel’s Digital Corpora is also looked into as a valuable source for setting up an experimental environment.

Chapter 3 is a technical review extending the literature review, first in order to discuss which platform is most suitable to use for the design of an application, followed by a review of the chosen platform’s current approaches by forensic tools and why they have flaws, also the Digital Forensics XML format and anti-forensic methods to overcome, as well as the most suitable programming languages to use and the best methods of development for this project.

Chapter 4 then introduces the design for the main application and schematics for the main requirements. The tool will hopefully be viable and efficient with all the main operations working. Research into the scientific method is also executed to draw out the metrics to be used.

Chapter 5 introduces the design for the chosen initial modules that will be implemented and Chapter 6 shows the scenario design that will be used and the experimental environment.

Chapter 7 implements the design into a final prototype from scratch. Samples of code and screenshots are provided to have a better overview of the prototype and its development. Are also shown the experimental environment implementation and other issues.
Chapter 8 shows the evaluation of the developed tool in two parts: first a validation type testing is performed to ensure the tools’ functionalities, and then experiments are run to evaluate the prototype and compare the results. Its functionalities are also compared with another open-source tool.

Chapter 9 finally concludes the research and compares the initial aim and objectives of the project with the final results. The difficulties encountered and overcome are also discussed, as well as project management and self-reflection. A possible transition to a future project is also envisaged.
2 Literature Review

2.1 Introduction

This chapter presents a literature review to have a better understanding of the important aspects of this project. The first section presents research into the general subject of Digital Forensics, and the current approaches and techniques used in today’s investigations, in order to understand the basic methods as well as the current challenges to overcome. Alongside this, input from the Scottish Crime and Drug Enforcement Agency (SCDEA) was gathered to reinforce the main requirements for a useful new forensics application. A comparison of traditional forensic methods and forensic triage is also carried out, to be able to evaluate which techniques are used in which investigative situation, and the reasons for using various methods. Research of currently available forensic tools is also included in the review. Finally, the legality, being a mandatory part of Digital Forensics, is reviewed to understand the effects of such methods, for example in a corporate environment. This last part includes a discussion around the Association of Chief Police Officers (ACPO) guidelines when dealing with electronic evidence.

2.2 The Digital Forensic Approach

A very concise, well written, and easy to understand definition of Digital Forensics can be found in the book “Digital Forensics for Legal Professionals” (Daniel & Daniel 2011, p. 3):

“The term forensics can be defined as the application of science to a matter of law. The most accepted definition of digital forensics comes from the definition of computer forensics: the collection, preservation, analysis, and presentation of electronic evidence for use in a legal matter using forensically sound and generally accepted processes, tools, and practices.”

In other words, Digital Forensics refers to the way electronic evidence is processed to be usable in a legal setting, for example to present in a court in front of an audience who does not have the same technical knowledge of a forensic investigator. In order to reach this goal, four important steps are used, each having its own techniques and following strict procedures: the Evidence Collection, Preservation, Analysis and finally the Presentation. It is important to understand such a concept, as digital evidence can easily be tampered with if a strict procedure is not followed, which will render it invalid to the eyes of the justice. It is therefore important to handle it in a correct way, and this is only one of the challenges present in Digital Forensics.
2.2.1 Evidence Collection

A concise definition of evidence collection would be “the acquisition of all information related to a given incident, in both electronic and hard formats” (Forte, 2003). In other words, the first step required to begin a digital forensic investigation is to gather information. This is also the most important step in regard to this project as it should be one of the main functions of the application.

From here, there are two ways the investigator can start the investigation. By hard-shutting down the machine, such as directly unplugging it, and then collecting the information using a chosen forensic tool; or else by doing a live forensic investigation on-scene with the system still running. However, if the latter is executed improperly the evidence may be corrupted, and because whatever is performed cannot be then reversed, this evidence could be refused in a legal setting (Kruse & Heiser, 2002). This collection step can raise some difficult issues for an investigator: for example the machine could have a very large storage size and it would take a very long time to obtain an image of it, thus losing a precious amount of time. There could also be many devices linked to this machine such as USB keys, other external hard drives or various pieces of hardware, but the investigator would first have to obtain an image of the drive to find this out. The tool used could also have compatibility problems with the Operating System, or the drive be fully encrypted, and live forensics may be deemed too large a risk to take in such conditions as volatile data can easily be tampered with and the examiner would need to have a very good reason for doing so. One solution in such a case would be to have information on whether any traces of possibly interesting evidence would be on this machine in the first place (Roussev & Quates, 2012). It might be more useful to know this type of information before spending a very long time investigating large digital devices which may lead to no evidence being found. Such method is known a Triaging and will be reviewed in later sections.

2.2.2 Preservation

Preserving evidence means keeping it safe. The victim’s or criminal’s own hard drives are not a safe place to investigate the evidence and the data needs to be copied to another safe medium, so the investigator can analyse it without the risk of modifying important information (Kornblum, 2002). It is also interesting to note the type of hardware that should be used when dealing with a hard-shutdown of the machine, or a live forensic investigation. When hard-shutting down the machine, attaching a hard drive is useful; however for live forensics, the “best method of extracting data from a machine is via the network connection”, and not via a device such as a USB drive as it can performs modifications to the system (Kornblum, 2002). The evidence collection and preservation seem to be the most delicate steps as the integrity of the evidence can easily be affected.
2.2.3 Analysis

Once the entire digital media has been collected and preserved, the examination can start. The investigator would need to locate potential evidence artefacts in the data gathered and is known as the examination process (US Department of Justice, 2004). Analysis therefore refers to interpreting the recovered data to put it in a logical and useful format that is useful for the investigation (US Department of Justice, 2004). If a different approach, as seen in the “Evidence Collection” section, could be taken, such as doing some forensic analysis to find traces of useful evidence before conducting a full and time consuming analysis on the collected data, for example using triaging methods, this would greatly reduce the work and time lost by an examiner.

Also, in the case of computer triage, standards for writing and reading digital forensics XML (DFXML) files could be used to improve even further the speed of investigations in general (S. Garfinkel, 2012), as such files could be read by different tools.

2.2.4 Presentation

This last step is generally performed in front of the audience, since the analysis stage has been completed. The evidence and findings of the analysis are presented (Carrier, 2002). This relies on the “Daubert Law”, which validates the evidence as relevant and reliable, as will be seen later in the legality section. This is an important stage, however presenting the evidence will not be the main focus in this project.

2.2.5 The Challenges to Overcome

Today’s Digital Forensics are being threatened in various ways, so it is important to know which challenges are there to overcome. The methods currently used are becoming “irrelevant” and Digital Forensics are “facing a crisis” (Garfinkel 2010, p. 66). The most important changes that Garfinkel sees coming in the field are very interesting as the list summarises some of the current needs for new approaches and methods.

As technology is evolving, so is the storage size of electronic devices. Capturing a forensic drive image from large devices is becoming impractical, and the time spent to process the data on this image can be even longer. This is also a hardware problem, as the use of embedded flash memory, Solid State Drives (SSD) and other hardware interfaces are becoming more common (S. L. Garfinkel, 2010), as these types of devices are easily and quickly erased by the user. Added to that, forensic tools need to adapt to this new technology: Operating Systems (OS) and file formats are evolving, in turn increasing the complexity of the tools (S. L. Garfinkel, 2010). This includes RAM investigation, which is difficult to implement in forensic applications, making it even more costly (Yadav, 2011). Today’s investigations are also taking significantly longer because investigators have to perform analysis on not only one machine with one hard drive disk, but usually many other devices as well. This is the only way to link the devices together to extract efficient evidence and a logical link between all
these devices (S. L. Garfinkel, 2010). This is rendered even more complicated including the fact that full disk encryption is a more common practice, even by the most typical users, as free programs such as TrueCrypt are very easy to use, well-known and easily accessible, and a disk cannot always be processed for that reason. (Casey, Fellows, Geiger, & Stellatos, 2011). Other problems such as Cloud Computing and legal challenges are also producing new challenges. Cloud Computing can prevent the retrieval of some pieces of data in some cases and can be a very complex procedure involving a lot of time and resources (Daryabar, Dehghantanha, Udzir, & Fazlida, 2013), and Legal challenges can limit the scope of Digital Forensics as well, due to the jurisdiction involved. For example, when dealing with an investigation within a corporation in the U.K., it is important to stay within the law in relation to the Data Protection Act, but it is also important to be certain that the seizing of multiple machines is needed, to avoid legal issues. All these challenges are faced by Digital Forensics investigators currently and is putting a large burden on law enforcement and. Other technical challenges also come into consideration which will be reviewed later in the technical review chapter.

2.3 Real World Requirements

A meeting and emails exchange with Mike Dickson, from the Scottish Crime and Drug Enforcement Agency, enlightened this research in terms of authentic digital investigations (Appendix 4 and 5). The main findings after an initial interview were that it seems even the Police are, still, after a couple of years, looking at methods to speed up the process of looking for evidence. More precisely, they have a real need for a “triage tool that will determine whether or not a computer would have data to be of interest in the first place”. This solution may help in deciding on a course of action to take, for example whether it would be necessary to run a full analysis on a particular machine or not. It seems such a method has not been implemented yet, or not in the way they expected as current tools do not always include enough configuration capabilities, do not find useful pieces of information, or do not assist the investigator as the tool is running (Dickson, 2010).

Other interesting issues were also raised, such as the conditions in which a raid is executed on a suspect. This included example scenarios, for example in the case of a person suspected of possessing Indecent Images of Children (IIOC); usually the agents will perform an assessment of all the electronic devices on-scene, and look for images, encrypted containers or encryption programs, and other types of digital files which contain suspicious keywords. Another example was in the case of a fraud, where this time the agents will look for evidence such as digital files containing bank related keywords in their name or contents. This kind of scenarios are helpful to understand what kind of evidence is useful to retrieve, and could also be used as experimental scenarios. It was also mentioned how important the analysis of volatile memory is becoming when it is possible, for example to find encryption keys stored within it. Also, depending on the investigation, typically it will be a person with little or no technical knowledge who would run a triage tool on the suspect’s machine, but will have the minimal training to do so. Therefore minimal technical interaction and decision making should be required for a successful acquisition of the data. Also, again depending on the
scenario, time will be an important parameter, so even if the suspect’s machine possesses a very large hard disk, it would be ideal to perform the investigation quickly, usually one hour. The “FieldSearch” and “COFEE” tools were also mentioned during the interview, and a copy of the “COFEE” application was acquired from SCDEA. Unfortunately, FieldSearch is available only to the law enforcement and could not be reviewed, even after trying to send emails to the company directly no version of the tool could be found to analyse. COFEE on the other hand provides a graphical user interface and is a wrapper of many different tools, and it is possible to configure it in advance in order to run it from a USB key directly on the suspect’s machine. However, the application itself seems rather confusing to use and again, it does not assist the investigator directly in knowing whether evidence is present, but rather focusses on extracting data such as full internet history logs, or decrypting passwords. It does however contain interesting functionality such as the analysis of volatile memory. Another open-source example of a triage tool is TriageIR, which is a wrapper of other tools including the Microsoft Sysinternals Suite and RegRipper. Again, the application presents the user with checkboxes to select, which will then enable or disable some of the utilities and commands run. It will then gather all the evidence in text files, and do not provide any alerts nor directly assist the investigator while it is running. However it provides a full analysis of the Windows registry and it is also possible to build a tree of all the folders present on the system, which are good additions.

2.4 Traditional vs. Triage Digital Forensics

As seen in the previous sections, traditional Digital Forensics is facing difficult challenges. The simple data acquisition and analysis of a hard drive can take a long time and can lead to no final result, and this method will probably even become irrelevant in the near future (Golden G. & Roussev, 2006). Looking forward, new techniques, methods and approaches have to be developed, and this also includes the tools used in Digital Forensics. A comparison study of traditional Digital Forensics and Triage Digital Forensics is conducted in the following section. So far, it is understood that the traditional forensic methods are used for the collection, preservation, analysis and presentation of evidence.

Triage is one of these new approaches to be considered. The term takes its origin from a French word, meaning “to separate out” or “action of sorting items according to quality” (Oxford University Press, 2014). A digital forensic triage is one of the techniques that seems to be very useful these days, particularly to deal with the expansion of storage. The main purpose of forensic triage is to be able to find directly the data of interest on a machine which may be more fully analysed later. The following definition from the journal “Digital Investigation 9” illustrates this idea very well:

“The fast initial screen of the acquired data whose purpose is to identify the most relevant artefacts and help build an overall understanding of the case” (Roussev & Quates, 2012)

In other words, a forensic triage tool could help determine if there is anything of interest to the investigator on a machine in the first place, greatly reducing the time of a full forensic investigation afterwards. Such a method could find traces of possible evidence on a computer,
such as folders containing encrypted data, or suspected services running, which would be an indication of suspicious activity and may allow for further investigation. Furthermore, the same conclusions are reached after interviews with Mike Dickson from SCDEA, as seen in the previous section. This shows that new triage tools are a necessity right now and new designs are being sought, even by the law enforcement. Although this kind of method would assume that the end user is not using advanced anti-forensics methods to hide evidence, it still has to be taken into consideration. Examples of current triage tools are:

- **“TriageIR”, “TR3Secure”, “Kludge forensics”**
  Open Source tools which are similar in terms of functionality and were reviewed in recent research (Shiaeles, Chryssanthou, & Katos, 2013).

- **“FieldSearch”** (Solutions, 2014)
  More difficult to get access to as restricted to law enforcement; can do operations such as web browsing history, recycle bin data, registry analysis and is multilingual.

- **“EnCase Portable”** (GuidanceSoftware, 2014)
  The portable version of one of the industry standard forensics tools available. It is also capable of doing triage, but costly.

These tools provide common triage capabilities, but do not always provide full customisation by a more advanced user. An ideal approach may be to design a way for both a technical and non-technical user to do modifications in a standard and more advanced way, in order to provide different functionalities based on the scenario.

### 2.5 Legality

#### 2.5.1 General Overview

An increasing number of counter measures such as Intrusion Detection Systems (IDS) are being implemented in today’s companies. However, not as many measures are taken for Incident Response, which means after an incident has already happened. In 2002, 73% of UK businesses affected by an incident took the decisions to forget about it and resume their normal activities (Sinangin, 2002). The main reasons behind this are their lack of knowledge, the cost impact, and the fact that digital forensics seem to be a concept accessible only to experts or the law enforcement. It is only in 2006 that it is shown that commercial organisations are starting and increasingly making use of forensic computing for investigating incidents, such as fraud, IP theft, money laundering, accessing and distribution of pornography, harassment and so on. This is interesting as it shows that adequate methodologies and tools are becoming a necessity in a corporate environment as well as for law enforcement (Haggerty & Taylor, 2006). In 2013, the information security arm of the Government Communications Headquarters (GCHQ) and the Council of Registered Ethical Security Testers (CREST), amongst others, launched two Incidence Response schemes to try to help targeted companies (Leyden, 2013).
The cost and time of an investigation in a corporation would greatly depend on the methodology and technique used to deal with a particular scenario. For example, if an employee is identified as doing doubtful actions on a machine, such as Cyberslacking (Vitak, Crouse, & LaRose, 2011), or worse, it may be overkill to seize the hard drive, and fully analyse it to find evidence. In this case, Dr. J. Haggerty and Dr. M. Taylor propose an investigation process for a corporate environment different from the conventional one. Defined as: identify the purpose of the investigation and setup the requirements, acquire the evidence, analyse it and collect the results (Haggerty & Taylor, 2006), which is different from the traditional process as the first step is to make decisions on what actions to take. This approach would greatly reduce costs and time-efficiency and proves that one general method cannot be used for all scenarios. The important step here, in relation to this project, is to identify the purpose of the investigation. Triaging, as seen in the previous section, would be a good way to give more information on the targeted machine, because it would be fast, and would not disrupt the company’s operations, which can make an investigator responsible for. For example, if law enforcement decides to seize multiple machines in a corporation and do not manage to find any traces of evidence, it would be a disaster for the company, thus it has to be avoided. Regarding the legal aspect, it is also important to understand the importance of the gathered evidence’s reliability. Evidence gathered from a Digital Forensics tool is qualified as scientific evidence and has to be relevant and reliable before being admissible and so settled by the judge in a “Daubert Hearing”, which evaluates the reliability of scientific evidence in this case, based on a set of rules and standards (Carrier, 2002). It is however important to note that this procedure should be about qualifying experts and not tools themselves. Simply showing evidence from renowned software does not prepare to testify. This issue of reliability is composed of four parts, as highlighted by Brian Carrier and illustrated in the figure below.

![Evidence Reliability](image.png)

Figure 1 - Evidence Reliability
2.5.2 ACPO Guidelines

Electronic evidence has become as important as hard evidence, and therefore must follow the same rules (ACPO, 2013). In this regard, it must comply with the Association of Chief Police Officers (ACPO) guidelines, in association with electronic evidence. Those guidelines follow four principles:

- **Principle 1**
  “No action taken by the law enforcement agencies or their agents should change data held on a computer or storage media which may subsequently be relied upon in court.”

- **Principle 2**
  “In circumstances where a person finds it necessary to access original data held on a computer or on storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.”

- **Principle 3**
  “An audit trail or other record of all processes applied to computer-based electronic evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.”

- **Principle 4**
  “The person in charge of the investigation (the case officer) has overall responsibility for ensuring that the law and these principles are adhered to.”

With the first principle in mind, it is logical that the first step for an investigator would be to take an image of the hard drive, or other media, in order to perform forensic analysis on it later. However, as it was mentioned in the previous sections, this is not always possible, and can take too much time. This makes principle 1 a technical challenge for triaging and must be taken into account in the design of any triage application. Other principles are less of a concern as they do not directly affects triage tools.

2.6 Digital Corpora

In order to be able to perform validation and evaluation, it is important to design scientific experiments. This implies that all experiments carried out, as it will be seen in the “Scientific Experiment” section, should be reproducible by (S. Garfinkel, Farrell, Roussev, & Dinolt, 2009) (Flandrin, 2012). But in order to do so, a set of standard digital files that will be selected for the experiments need to be used; Simson Garfinkel’s digital corpora provides realistic standard sets of files which can be used. A set containing an important number of digital files is available which open access and could be used to run such experiments. It would be for example a good idea to set up a virtual environment containing these files and run the experiments on them, as it would represent best a suspect’s machine where the triage
tool could be run. Recent research at Napier University shows that creating manually such images containing corpora files is a good choice for validating tools, but may however reveal to be difficult, time consuming, and the files used are not always varied (Russell, Macfarlane, & Ludwiniak, 2012). It should however be an adequate method in this case due to the size of the project.

### 2.7 Conclusions

From this literature review, a better understanding of digital forensics has been drawn including the general approach to collecting, preserving, analysing and presenting evidence (Carrier, 2002; Forte, 2003; S. Garfinkel, 2012; Justice, 2004; Kornblum, 2002; Kruse & Heiser, 2002); The current challenges to overcome during forensic investigations have been highlighted, namely the increased storage sizes, use of new types of media such as SSDs, an increased complexity of the tools including RAM analysis, increasingly long examinations due to the different linked devices on-scene, full-disk encryption and also cloud computing (Casey et al., 2011; Daryabar et al., 2013; S. L. Garfinkel, 2010; Yadav, 2011). Real-world requirements have also been researched, thanks to interviews carried out with practitioners from SCDEA, such as the conditions of a typical, hour-long, raid; the fact that on-scene investigators do not necessarily have the technical knowledge of the tools they use; also that a tool that would tell Scene of Crime officers if evidence is present on a device may be useful or not. Meetings with personnel from SCDEA also gave good feedback on the currently available forensic tools and in particular their lack of direct assistance given to the investigator on-scene in making decisions (Dickson, 2010). The differences between the traditional approach and triage techniques also have been discussed (Golden G. & Roussev, 2006; Roussev & Quates, 2012), which led to the similar conclusions taken from interactions with SCDEA, that a tool that would directly assist the investigator in knowing if evidence is present on a digital device or not at the scene could help reduce the amount of digital media to be acquired and analysed later in the investigation. A general understanding of the legality associated with investigation has also been reviewed, looking at a corporate environment (Haggerty & Taylor, 2006; Leyden, 2013; Sinangin, 2002; Vitak et al., 2011), including the Daubert Hearing (Carrier, 2002) and the ACPO guidelines (ACPO, 2013) which relate to law enforcement investigations. Finally, digital corpora has also been looked into (S. Garfinkel et al., 2009), which could be useful for creating good scientific experiments, which would allow consistent results for an evaluation of any tool created (Flandrin, 2012). Overall, it can be concluded that the current challenges and needs in the real world seem often to push forensics towards using a triage approach, which is becoming vital as traditional techniques become less and less effective in certain scenarios. The next chapter extends this literature review to add a more technical approach to some of the issues.
3 Technical Review

3.1 Introduction

This chapter presents a technical review, following from the literature review, in order to have a better understanding of the technical aspects of digital forensics relating to the project. Thus, the first section provides research and comparisons of the most popular Operating Systems related to Digital Forensics investigations, including for example the architecture used (32-bit vs. 64-bit computing), in order to have a clear focus towards the design and implementation of a useful forensic tool. After that, the role that the selected Operating System plays in Digital Forensics will be evaluated, as well as the current triage methods used, including a review of a number of current triage tools. This includes evaluation of what they attempt to extract, and a discussion on which artefacts are important to consider in different scenarios. The Digital Forensics XML format is also looked into as a standard for forensic data representation (S. Garfinkel, 2012). Then, the most suitable programming languages to use when developing forensic tools are discussed, as well as possible methods of development, such as Agile.

3.2 The Different Platforms

3.2.1 Introduction

In order to have a clear focus in designing a technical approach to deal with some of the challenges that Digital Forensics practitioners are facing, it is first of all important to know which platforms would be the most affected. This depends on several factors, but the main one in this case would be the popularity of the Operating System (OS) according to statistics. Amongst the public, three Operating Systems are the most used: Windows, OSX and Linux (NetMarketShare, 2013). Even though the list of available OS is rather long, other OS than these three mentioned here are not as relevant as they do not have a huge number of people using them on a daily basis, so would not be encountered as often by investigators.

3.2.2 Windows, OSX, and Linux

Microsoft Windows is the most widely used operating system containing a Graphical User Interface (GUI). Its current version is Windows 8.1 but Windows 7 is still the most used for now (Wikipedia, 2014). This makes it most suitable for the purpose of this project as it is most likely a suspect will be using this system. OSX is another operating system using a graphical interface, based on UNIX and sold by Apple. Its current version is Mavericks and is free to use, unlike its previous versions, which makes it more accessible (Apple, 2014). However, it is not nearly as popular as Windows. Linux is a UNIX-like operating system which was developed as a Free Open Source Software (FOSS). A tremendous amount of different distributions exists, either with a graphical user interface or not and is the most commonly used operating system on servers (W3techs, 2014).
All three operating systems can be accessed fairly easily for the scope of this project, and all provide a good environment for developing, however the Windows OS seems more appropriate for this project because of its popularity (as discussed in section 3.2.4) and ease of development. It will also be easier to setup a virtual image snapshot for experimental purposes, representing a typical user.

### 3.2.3 64-bit vs. 32-bit Computing

Another important point to take into consideration is which architecture to use between 64-bit and 32-bit computing. The latter can handle memory much less effectively than the former, and this has to be taken into account if designing a program (Microsoft, 2013). For example, a 32-bit application can run on a 64-bit system, but the opposite is not true. Different versions of a same piece software must then be created, or one version should work on both architectures.

### 3.2.4 Statistics of Popularity

One of the main factor to take into consideration when deciding which platform to focus on when dealing with Digital Forensics is its popularity. For showing this, two sources are used: one table from the W3School’s log files, collected over a period of nine years (the year 2013 is selected) (Table 1), and one pie chart from the Net Market Share website, giving statistics for Internet Technologies (Figure 2).

<table>
<thead>
<tr>
<th>2013</th>
<th>Win8</th>
<th>Win7</th>
<th>Vista</th>
<th>NT*</th>
<th>WinXP</th>
<th>Linux</th>
<th>Mac</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>September</td>
<td>10.2%</td>
<td>56.8%</td>
<td>1.6%</td>
<td>0.4%</td>
<td>13.5%</td>
<td>4.8%</td>
<td>9.3%</td>
<td>3.3%</td>
</tr>
<tr>
<td>August</td>
<td>9.6%</td>
<td>55.9%</td>
<td>1.7%</td>
<td>0.4%</td>
<td>14.7%</td>
<td>5.0%</td>
<td>9.2%</td>
<td>3.4%</td>
</tr>
<tr>
<td>July</td>
<td>9.0%</td>
<td>56.2%</td>
<td>1.8%</td>
<td>0.4%</td>
<td>15.8%</td>
<td>4.9%</td>
<td>8.7%</td>
<td>3.2%</td>
</tr>
<tr>
<td>June</td>
<td>8.6%</td>
<td>56.3%</td>
<td>2.0%</td>
<td>0.4%</td>
<td>15.4%</td>
<td>4.9%</td>
<td>9.1%</td>
<td>3.2%</td>
</tr>
<tr>
<td>May</td>
<td>7.9%</td>
<td>56.4%</td>
<td>2.1%</td>
<td>0.4%</td>
<td>15.7%</td>
<td>4.9%</td>
<td>9.7%</td>
<td>2.6%</td>
</tr>
<tr>
<td>April</td>
<td>7.3%</td>
<td>56.4%</td>
<td>2.2%</td>
<td>0.4%</td>
<td>16.4%</td>
<td>4.8%</td>
<td>9.7%</td>
<td>2.2%</td>
</tr>
<tr>
<td>March</td>
<td>6.7%</td>
<td>55.9%</td>
<td>2.4%</td>
<td>0.4%</td>
<td>17.6%</td>
<td>4.7%</td>
<td>9.5%</td>
<td>2.3%</td>
</tr>
<tr>
<td>February</td>
<td>5.7%</td>
<td>55.3%</td>
<td>2.4%</td>
<td>0.4%</td>
<td>19.1%</td>
<td>4.8%</td>
<td>9.6%</td>
<td>2.2%</td>
</tr>
<tr>
<td>January</td>
<td>4.8%</td>
<td>55.3%</td>
<td>2.6%</td>
<td>0.5%</td>
<td>19.9%</td>
<td>4.8%</td>
<td>9.3%</td>
<td>2.2%</td>
</tr>
</tbody>
</table>

Table 1 - W3School's OS Platform Trends for 2013 (W3SCHOOL, 2013)
Both Table 1 and Figure 1 show different numbers, but overall it is clear that the most popular Operating System used is Windows, Windows 7 being the most popular version currently. Statistics are not an exact science, however if used to compare two different platforms usually widely used such as Windows and Linux, they show that Windows plays a much larger role in general everyday usage. Furthermore, in a corporate environment, the Windows Operating System is the most utilised one, because it is much easier for the general user. Interestingly enough, it seems that Windows XP is losing in popularity and both Windows 7 and 8 are becoming more popular, mainly because older versions are losing support, as it has been the case recently with Windows XP (Microsoft, 2013). Windows Vista is still at a very low popularity rate. From there, it is safe to say that Windows should be the Operating System of choice to continue the technical review, whether it is for a business environment or for individuals.

3.3 Windows in Digital Forensics

3.3.1 Introduction

The Windows Operating System being the chosen platform to focus on for the rest of this technical review, it is now necessary to look at the role the Windows OS has been playing in Digital Forensics the past years, compared to how it is performing now. Then, a list of the most relevant Windows artefacts which might be gathered during an investigation, in correlation with the previous literature review and the already mentioned triage tools, will be made, and a carefully selected sample will be analysed in more details. The access through a Linux live CD is also discussed as it is not always possible to access a Windows session via the general user login session, and therefore may need to be bypassed in some way.
3.3.2 Role

In the recent past (1999 – 2007), digital forensics were at what Garfinkel defines as a “Golden Age” (S. L. Garfinkel, 2010). This is argued by giving a few characterisations on why this was the case:

Windows XP was the most popular Operating System at this time: Apple and Linux devices were not as used as they are today, and this was making the life of investigators much easier as they typically only needed to worry only about one OS to work on. Just a few files formats were used: the scope and speed of Digital Forensics were greatly reduced by the fact that only a few extensions such as Microsoft Office documents, JPEG images or AVI videos were used, as opposed as today’s where hundreds of different formats can be found on one machine. Most cases were composed of a single machine for a single individual: in today’s investigations, it is important to link multiple external devices (if any) as quickly as possible to reinforce the evidence, whereas in the past, investigation would usually be performed on a single machine (S. L. Garfinkel, 2010).

The forensic investigations were relatively simple at this time, but in the last few years things have completely changed. Operating Systems have evolved and many challenges appeared, as discussed in section 2.2.5. The Windows platform obviously plays an important part, as most actions likely to be investigated are perpetrated from or to a Windows machine. Furthermore, basic triaging processes in general rely on the fact that the end user is not using advanced anti-forensics methods (Dickson, 2010), also being developed and improved for Windows, such as encryption, or even techniques targeting directly certain forensic tools. A good example of that is the DECAF tool originally created as a counter to the Windows forensics tool COFEE, which looks for the forensic tool’s signature every time a USB device is plugged in, and then reacts to counter it, for example by deleting the report files. This shows that even the most reliable tools can be countered easily.

3.3.3 Current Triage Methods

In order to understand how triage is currently performed, two triage tools used at the moment will be reviewed, namely COFEE and TriageIR. These were discussed in the literature review (section 2.3). In this chapter, a deeper review is performed, discussing functionality, advantages, drawbacks, usefulness to practitioners and what they produce. The purpose of this is to understand how they work and what improvements would be useful.

On first run and as seen in the first screenshot below, COFEE presents the user with an interface where various choices are possible. The tool seems very configurable, which is good as this was highlighted as a key need or law enforcement practitioners. It is necessary to first select the USB device where the tool will be run from and the output extracted to, the mode, which provides a list of scenarios that are already prepared; and case notes - including the case ID or description. It is also possible to directly select the tools to use in the configuration, as seen in the second screenshot.
A total of 154 tools seem to be present, ready to be added to the current configuration. It is also possible to save the current configuration to a profile, which is a useful addition. However, it seems that all the utilities present are all responsible for “showing, listing, extracting” fixed pieces of evidence from the system; for example it may display the Internet Explorer cache entry, or show the Windows licence key. These tools seem to be rather basic which is quite surprising coming from a professional tool such as this one. It does however contain functionality which allows the user to customise, adding other tools, and to generate and view various reports created. These are useful features, especially that it is possible for the user to add their own tools and customise report formats. In the literature review, it was mentioned that the tool looks quite confusing at first, but after interacting and using the tool, it became more straightforward to use. Finally, even if the overall functionality of the application is good, it still does not specifically tell the investigator if the machine is suspicious enough to be seized in a short period of time. Instead, it creates reports which
should have to be analysed later by someone with a good level of knowledge and only then, decisions could be taken on the likelihood of useful evidence being present. It would be better to have a system of alerts that could be generated while the tool is running, which could quickly determine if further investigation was needed and to help decide quickly to acquire the media or not.

The TriageIR tool is comparable to COFEE, in that it presents the user with a list of utilities to be run from a user interface, as seen in the first screenshot below. It provides the user a configurable GUI interface, which can also be modified using an INI file. This is a good idea as it enables users with less technical knowledge to use the interface, and more advanced and technically comfortable users to directly modify the INI files, based on their needs. The triage tool however does not include any report creator or specific viewer as COFEE does. Instead, it simply runs the utilities selected and store the evidence in text files, as seen in the second screenshot below.

Overall, TriageIR offers some nice functionality similar to the COFEE tool, due to the fact that it uses the Microsoft Sysinternals Suite utilities, and also other tools such as RegRipper. However, even though it does provide useful capabilities such as the full analysis of the Windows registry – which is used as a storage for the configuration information of the system and its applications (H. Williams, 2007), and can store from simple configuration details to account passwords or encryption keys, which are very important to digital forensic
investigators (Dolan-Gavitt, 2008) – it does not directly assist the investigator in determining if evidence is present on a digital device at the scene. It seems to be a less advanced and open-source version of COFEE rather than a completely different tool.

Generally, it can be concluded from this review of COFEE and TriageIR that they offer good configurable functionality and set of tools, but they will only look for fixed, standard, pieces of evidence which may or may not be relevant to a specific case or scenario. Such methods show the flaws of current approaches, regarding how evidence is treated by current forensic triage tools. They only help to find specific pieces of evidence, and do not directly help the investigator to make decisions for specific situations. This is a problem that could be solved, for example by adding a system of alerts and binary decision displayed to the user while the tool is running, and offering a set of modules that could be modified or added easily that would look for suspicious activity rather than particular pieces of information. Moreover, they typically help to find evidence on a computer to help solve a crime against a particular individual, but not against computers (S. L. Garfinkel, 2010). They also do not seem to use any particular output format, which can restrict portability between different tools.

3.3.4 Improvements on Current Triage Methods

As seen in the literature review, it is more efficient to check for possible relevant artefacts in the first place, in order to decide on a further course of action in digital forensics triage (Roussev & Quates, 2012). It is also possible to extract relatively quickly an important sum of information from a targeted machine. However, this does not actively help in making decisions for the rest of the investigation, and a full use of a triage tool such as TriageIR gives too much output, and this output is not even formatted to any particular standard.

The next step would then be to directly process the selected pieces of information when running the program. This would be decided beforehand by the user, for example by using a configuration file. Depending on the needs, some information may be displayed on-scene, and some stored for later analysis. This process is made easier as it would not need to be completely forensically sound, being run on an “active but idle” system (Dickson, 2010). The minimum requirement would still be following the ACPO guidelines, which is to avoid writing files to the system and access data in read-only mode (ACPO, 2013; Dickson, 2010). From the interviews with Mike Dickson, depending on the various triage related scenarios, the following functionalities revealed to be of utmost importance:

- Researching files by hash values:
  Hashing is one of the most important tool in Digital Forensics, and can significantly increase the speed of an investigation (Roussev, 2009). However, it can take a relatively long time to do a hash of all files of a system. As seen in the literature review, time is an important factor in triaging, and therefore a solution would be to do a hash of the files of interest to an investigator. The configuration file could also include a set of extensions such as “jpg”, “avi” or “doc”, in order to filter the most relevant files, so it would speed up the process even more.
• Finding files using a text-based search feature:
  For example when looking for specific file names, contents or even when going through a list of installed programs in case hacking or encryption tools are installed, which would alert the investigator that there are traces of suspicious activity; or even web browsing activity, for example in the case of a missing person where such logs could help give clues on where a person could be and the reason they have been missing.

• Looking for possibly encrypted files:
  This may be more complicated, but different techniques exist with varying success rates depending on complexity of technique. One way of doing this is to look for even distribution of bytes value in a file (Dickson, 2010). This is possible due to the fact that encrypted files have an evenly distribution of data; therefore it is possible, using counters, to compared the highest number of a particular byte value to the lowest number of another value, and generate a statistic on whether the file is encrypted. After discussing these techniques with Philip Penrose (Appendix 6), a PhD student at Napier University working on a neural network capable of detecting encrypted files, and implementing a C# version of it to run some quick experiments on different encrypted files, the conclusion was that the algorithm would indeed be capable of detecting encryption, but not for files small in sizes such as 4KB or 8KB. Therefore this technique of quick crypto check (QCC) would be true for large files only, increasingly useful with larger files. The QCC technique should therefore be used only for files with a certain size limit, for example 5MB.

3.3.5 Anti-Forensics

Considering the time allocated to this project, it is important to note that limited research was carried out into countering anti-forensic methods. Instead, only encryption was focused on as it is a common method of hiding data, and a fairly straightforward way of detecting it was found, whether it is just an installed program or and encrypted container. The method discussed in the section above was designed in conjunction with Philip Penrose and Mike Dickson and will be incorporated into the prototype tool.

3.3.6 Accessing OS via a Linux Live CD

In some cases, the Windows machine will be password protected. However, this is easily bypassed in various ways for any version of Windows as shown in the massive amount of articles and videos on the internet when running a quick search. One common way of doing so is by using live CDs. They are used to directly run an Operating System from a CD and are usually Linux based, and do not install anything on the machine it is run on (Smith, 2007). Such a scenario would require the tool to be able to run on Linux as well as directly on Windows, so there would be no issues of compatibility when deploying the tool on-scene.
3.4 The Digital Forensics XML Format

The Digital Forensics XML format was created as a standard for representing forensic data in order to allow interoperability between forensic tools (S. Garfinkel, 2012). However, the project has been stopped for now due to a lack of funding, even though it is very aspiring (S. L. Garfinkel, 2012). The scope of this project does allow the implementation of this but is definitely something worth considering for future work.

3.5 Language Choice and Method of Development

For the scope of this project, an appropriate language to use might be Python, as it possesses a lot of advantages: it is easy to use, there are a lot of libraries related to forensics that can be called for added functionalities, and it can be run on any of the three operating systems presented earlier, in other words, it is cross-platform. It is also regarded to be an easy language to learn but very powerful, and good to use for quick prototyping (pySchools, 2014). For what concerns the method of development, the Agile methodology seems the most suitable as the process is fairly straightforward and the development simple: it involves doing testing every so often and small changes in a short period of time, which is perfect for the scope of this project (Pressman, 2010). The Waterfall model was also considered, but a disadvantage with that one is the lack of flexibility in the development process, meaning there would be a lot of documentation to do before the actual coding and testing and for every modification (Sommerville, 2011).

3.6 Conclusions

From this technical review, the three most widely used platforms were compared and out of them, Windows was the chosen one to use as the main test platform, because of its popularity (NetMarketShare, 2013; W3SCHOOL, 2013), and the important role it has been playing in Digital Forensics in the recent years; namely its increase in complexity over the years (S. L. Garfinkel, 2010). There is also the fact that quick triage methods imply that advanced forensic methods are not used. After an analysis of the current triage tools’ was performed, including a reading comparing three open-source tools and their functionalities (Shiaeles et al., 2013) showing the flows of current methods not directly assisting the investigator using them, but rather profiling the devices. From there, improvements of current triage methods were proposed, for example a way to adapt the tools in order to find if evidence is present or not on a digital device (Dickson, 2010; Roussev & Quates, 2012), and three techniques were highlighted towards that. Anti-forensics methods research was limited as to follow the scope of the project. Finally, Python was chosen as the language to use for development due to the fact that it is easy to learn and powerful, making it good for quick prototyping. It is also cross-platform, which would become important if a Windows machine would be password-protected and access through a Linux live CD would be necessary (section 3.3.6). Overall, functionalities for a triage tool are being drawn out, keeping in mind that current triage tools are not always assisting the investigator on-scene but rather just for collecting important data for later analysis. The use of a standard DFXML output is also considered but will not necessarily be included in this project, but rather for future work.
4 Triage Tool Design

4.1 Requirements

From the literature and technical reviews, it has been shown that triage tools usually search for fixed pieces of information and store them for later analysis, or they do not display this information quickly enough for the investigator to make decisions on-scene, or that there is not enough customisation (Dickson, 2010; Shiaeles et al., 2013). For example, the investigator would run the tool from a USB key, and then wait until the tool has processed the information on a machine before being able to make any further decisions. Therefore, instead of designing a tool which follows the same characteristics, a new tool should possibly have the ability to use different configuration profiles which could be useful depending on the scenario being investigated. The tool could also trigger alerts during the operation. This will be useful as various capabilities can be added and the investigator will know as fast as possible what they will need to do on-scene. Before the user goes on-scene and start the tool, they will have to configure it in order to gather the pieces of evidence they will need for a particular scenario. The application will need to be able to adapt itself to the configuration profiles the investigator will decide to use. As it has been seen in the literature review following interviews with SCDEA personnel, this should be either set by a person with minimal training, therefore using already-defined scenarios, or by a more advanced user able to create custom profiles directly by modifying the configuration files (Dickson, 2010).

The main part of the application will be started from a USB drive in Windows command prompt which, when started, should have access to the drives of the system it is run on, either taking input from the user for deciding which profile to use or start the tool directly using a profile selected in advance. The purpose is to have a minimum interaction between the tool and the user. Finally, it will process the information on the machine, keeping logs, storing and displaying important pieces of information that could help the investigator deciding of a further course of action. The application will then call different modules depending on which profile was selected, allowing the creation or adaptation of new functionalities, and start the triage process. If possible, the tool should run in memory to allow the USB drive to be removed while the application is still running, allowing the investigator to plug the USB into other devices if need be. As a triage tool, it should also be fast (Roussev & Quates, 2012), for example one hour as outlined by the real-world requirements discussed in section 2.3.
4.2 Experimental Design

4.2.1 The Scientific Method

The scientific method is the process for finding answers to the questions posed in a scientific environment and is composed of 6 steps (Dodig-Crnkovic, 2002): ask the question using the already existing knowledge and observations, propose an hypothesis, make predictions, test the hypothesis in an experiment, and if the experiment is successful, new theories can be achieved and/or old ones confirmed. Such experiments have to be reproducible and therefore can be performed over and over as much as is possible (Mocas, 2004). This means other researchers can also reproduce the experiments, and possibly build on them. As discussed in the literature review, the use of digital corpora can help with this for digital forensic tool experimentation (S. Garfinkel et al., 2009). The following diagram shows the process of iterating the experiments by taking scenarios, data and various outputs such as configurable parameters, including triage tool settings and configuration, in order to get the outputs, such as the triage results, including alerts and stored logs, and the metrics, such as the execution time and correctness. This is explained in more details in the next section.

![Figure 3 - Overview Experimental Design](image_url)

In the case of designing a methodology for a digital forensics tool evaluation, research has been fulfilled and a useful method such as this one can be used: define the requirements of the tool, as it has been discussed in the previous section; define how the application is evaluated; and then, for example by using the digital corpora files, run experiments multiple times and generate, analyse and evaluate the results of certain metrics which have been recorded (Beckett & Slay, 2007; Clayton, 2012). The experiments should also be documented precisely and thoroughly so they can be repeated if need be.
4.2.2 Experimental Conditions

As Figure 3 shows an overview of the experimental design, the metrics to be used will be composed of three parts: the scenarios that can represent a real-world environment as close as possible; the inputs for the experiments and the outputs. The scenario defines the evidence files to be discovered by the tool in a given context, therefore the tool will be run multiple times on each scenario but using a different configuration. It could be argued that the scenarios are part of the inputs, however they are more used as the base environment for each experiment and it is preferable to see the difference here. The inputs to the experiments are the triage tool settings, such as the choice of modules to run, and other configuration choices such as particular file extensions to check and so on. Finally, the outputs for each experiment consist of the overall time it took the application to complete its operation but also the time it took the application to find the first piece of evidence in each module, because it was discussed in the literature review that a typical investigation should take less than an hour; the hits made, which are the alerts generated; and whether they are correct or not which seem to be the standard metrics to use in such circumstances after reviewing other papers (Beckett & Slay, 2007; Clayton, 2012).
5 Modules Design

5.1 Introduction

It has been decided based on discussion in the previous chapters that the following three triage modules were needed for a prototype triage tool: files by hash value, files by name or keyword contents, and encryption detection (Dickson, 2010) (section 3.3.4). Recent research conducted by Bruce Ramsay at Napier University, who was also working with the Police, shows that triage modules can be separated into three categories: time-based information, text-based search and hash-based search (Ramsay, 2014). It is however not relevant to do it the same way here as time is a factor in all of these modules. However, it is a good idea to combine the files by name and files by keyword search modules together into one text-based search module. Therefore the three triage modules presented earlier can be reduce to three as follows: files by hash value, encryption detection and text-based search, because of research conducted in the field showing that finding files by hash values is a standard way of identifying known pieces of data (Roussev, 2009), that encryption detection would be the only anti-forensic method part of the scope in this project (section 3.3.5), and that text-based search can be used in many situations, from searching files by name or contents to recovering internet history logs (Dickson, 2010; Ramsay, 2014). Each of these modules is introduced and described in the following sections. Also diagrams and pseudo code are used for better understanding of the design of the processes involved.

5.2 Module 1: Files by Hash Value

5.2.1 Design

The first module presented here is responsible for calculating the hash value of the files on the media to be investigated in order to find evidence of bad files. While the main function of the prototype will be to select the media to work on and navigate through all the files, this module will read each file and calculate their hash value. It will also read a text file containing a list of known bad hashes prepared in advance by the investigator. It is important to note that only the file extensions chosen in the configuration file by the investigator prior to running the tool are used, as it has been seen in the technical review that filtering for extensions would speed up the process, and is also easier to implement for a quick prototype. Future work however might include a functionality that will check for magic number in the files, in case a suspect decides to change the extensions. Once this is executed, the two inputs are compared and if a hash value calculated is present in the known bad hashes list, it will display an alert to the investigator as well as storing the concerned file information. Figure 4 illustrates this, and Figure 5 provides pseudo code:
Figure 4 - Files by Hash Value Design Diagram

Figure 5 - Files by Hash Value Pseudo code
5.3 Module 2: Encryption Detection

5.3.1 Design

This module is to give hints to the investigator on whether a file might be encrypted. As seen in the literature review, encrypting files may be information on suspicious activity and has to be reported to the investigator on-scene. The module works as follows: once the current processed file is selected, the module will read its path and get its file size, as only file of sizes superior to the one decided by the investigator will be processed. If the file size is valid, it will then be processed using a “Quick Crypto Check” function reading the file byte by byte and return a result in percentage. If this percentage is higher than the threshold set by the investigator, it means that the file is probably encrypted. Figure 6 illustrates this, and Figure 7 provides pseudo code:

Figure 6 - Encryption Detection Design Diagram
5.4 Module 3: Text-Based Searching

5.4.1 Design

The idea behind this module is to regroup the user of keywords to search particular strings in the system. This includes for examples the search of file names, particular keywords within the files, or web history. The module works simply by reading an input file and looking for a particular string within it or in its name. If one or more of the included keywords appear, an alert will be displayed and the file information logged. As seen in the literature review, the triage method should be quick, therefore as soon as a string has been found, the module will stop looking for more keywords for one particular file and start reading the next file. Figure 8 illustrates this, and Figure 9 provides pseudo code:
Figure 8 - Text Based Searching Design Diagram

Figure 9 - Text Based Searching Pseudo Code
5.5 Method of Storage and Profile Structure

The alerts, as well as being displayed on-screen during the triage process, will be stored in text files with relevant names. These files will contain the file name and path so the forensic analyst will be able to spot quickly where the suspicious files are located on disk, thus saving a precious amount of time. In the literature review, the Digital Forensics XML format was mentioned, however the time allocated to this project does not allow to apply that to this tool. Future work however might implement it. A profile folder will automatically be created each time the tool is run. The name of the folder will be composed of the date and time. However, user input could also be added to give a specific name easily. Further improvements on that will be mentioned as possible future improvements to the tool.
6 Experimental Scenario Design

6.1 Introduction

This section describes the experimental design for the triage tool. This is based on the interview and research performed in the literature review, as well as research into the technical review and reflect real-world examples which are very valuable (Dickson, 2010; Ramsay, 2014). The designed prototype triage tool will be compared with another open-source tool. These scenarios provide different experimental conditions to use when running experiments for the applications, as seen in the experimental design section.

6.2 Scenarios

6.2.1 Scenario 1: Suspected Paedophile

It was seen in the literature review (section 2.3), after interviews with practitioners from SCDEA, that one of the possible scenario was the possession of IIOC. Usually, the particular pieces of evidence to find in this case are images, encrypted containers or encryption programs, and other digital files containing suspicious keywords in their name or contents.

Therefore, for this scenario, dummy pictures should be stored within the testing image, and some of their hash stored in a file containing other bad file hashes. An encryption program should be installed and different encrypted containers created. Finally, different text files should be added and some suspicious keywords used in their name or contents. This scenario is good to run the three modules of the prototype tool in parallel.

In order for the experiments to be successful, the tool should discover the correct pictures and documents amongst a set of other standard ones. The prototype should also find and alert on the encrypted containers. The program should clearly display feedback to alert the user. Along with displaying those alerts, it should also log them to make the work of future examiners quicker.

6.2.2 Scenario 2: Fraud Investigation

As seen in the literature review (section 2.3), this was another scenario mentioned following the interviews with SCDEA. In this case, some of the relevant pieces of evidence that could be retrieved concern digital documents that would contain suspicious keywords in their name or contents, for example banking information.

Therefore, dummy documents can be stored within the testing image, with some of them containing a set of suspicious keywords. This scenario has the purpose of testing if a module running alone affects the performance of the prototype tool. In order for the test to be successful, the tool should discover the correct document amongst a set of other standard ones. Along with displaying those alerts, it should also log them to make the work of future examiners quicker.
6.3 Experimental Environment

The best way to run the experiments detailed in the experimental design section is to use a virtual machine. This virtual machine could be hosted on a cloud platform, such as Napier University’s. The advantages of running the experimental environment using a cloud platform are that there is almost unlimited storage, it is possible to do backup and recoveries easily, and it can be deployed quickly (Apostu, Puican, Ularu, Suciu, & Todoran, 2013). It is also possible to run multiple machines in parallel, which would greatly reduce the time spent running the experiments. However there are also some drawbacks, such as the fact that technical issues can arise, including possible downtime or connectivity issues (Apostu et al., 2013). In the technical review, it has already been decided that the operating system to use would be Windows 7 64-bit (NetMarketShare, 2013; W3SCHOOL, 2013), and in the previous section, two scenarios were detailed. A proper experiment would then be to have three snapshots of Windows 7 virtual images containing different sets of digital corpora files (S. Garfinkel et al., 2009) related to each scenario. It has been decided that the tool should be run ten times on each snapshot using different adequate settings, for the reasons that each iteration can take a long time, and that it should be sufficient to be confident in the results as they should not all be that different. The following table illustrates this idea more clearly:

<table>
<thead>
<tr>
<th><strong>Experimental Setup</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario</strong></td>
</tr>
<tr>
<td>Scenario 1: Suspected Paedophile</td>
</tr>
<tr>
<td>Scenario 2: Fraud Investigation</td>
</tr>
</tbody>
</table>

Table 2 - Experimental Environment Setup

6.4 Conclusions

Those two scenarios are taken from the real world and so are very realistic and good to use for such a triage application, because they show well what the tool should find and which modules to use. They also all give a good set of evidence to find and all take the time factor into consideration, which is an important metric to use during the evaluation when comparing the tools in this case. The experimental environment is then set up according to those scenarios and the experimental design section previously discussed.
7 Implementation

7.1 Introduction

In the previous chapters, the design for a prototype triage tool and the reasons behind that have been presented. In this chapter, it is described how the design was implemented and samples of code are also provided. This was executed following the Agile development methodology as discussed in the technical review, by performing short developments and testing. Four components have to be implemented: the main program window and configuration file, which will be used to start other modules; the files by hash value module; the encryption detection module; and finally the text-based searching module. The implementation of the experimental environment is also described as it is a tedious and important process required for experimental purposes. The tool implementation is described first, followed by the experimental environment, and finally a summary of issues encountered during the overall implementation is presented. During the implementation, the default Python IDLE editor was used for the code and a simple text editor called Notepad++ for the configuration files.

7.2 Tool Implementation

7.2.1 Main Window and Configuration Module and File

![Figure 10 - Main Window Folder Structure](image)

The main window for the application is responsible for taking user’s input and calling the triage modules as well as gathering the first pieces of output such as general information of the system using the windows command “systeminfo”. The “cfg” and “output” folders respectively serve the purpose of holding the scripts responsible of loading the user-defined triage modules in the “cfg.ini” file and holding the output made by the application.
The first part of the code shown in Figure 11 is the general setup of modules and global variables used. The only non-native library used here is the win32api (or pywin32) and is useful for retrieving a list of the currently mounted drives that the user can select. The imported “modules” library is in fact another script part of the tool which is located in a different folder and is called when the triage process is ready to start. Once the Windows OS has been detected, the drives are listed and the user chooses one of them (Figure 12); finally the application reads the modules.py script (Figure 13) to know which modules have to be loaded using the “cfg.ini” file (Figure 14). After the user has confirmed the modules to load, the triage process starts in new windows as seen in the triage tool design section.

To add other modules, it is necessary to look at the overall folder structure as seen in the screenshot below. The python_triage folder contains the main python script responsible for running the main screen and calling the modules, which was discussed above. The other folders are modules associated with the triage tool; therefore, if other modules were to be created, new folders would be added there.
Figure 12 - Main Window Code part 2

```
# Main Window Code Part 2

getOS();
raw_input()
```

Figure 13 - Module Loader Code

```
# Description: Module loader for the python triage tool
# Author: Charley Célice <celicecharly@gmail.com>
# http://nevermore.turlorkh.net

import ConfigParser
import subprocess

# Names of modules to match with the .cfg.ini file
Config = ConfigParser.ConfigParser()
_loadedModules = ["encryptionDetection", "filesByHash", "textBased"]

# Configuration section to read
Config.read("./cfg.ini")
Config.sections()

ConfSectionKey = ConfigSectionKey(section):
    def __init__(section):
        self.section = section

    def getOptions(self, section, option):
        if option in options:
            try:
                return dict[option] = Config.get(section, option)
                dict[option] = dict.get(section, option) - 1
                print("Warning: option: {} is not found in [{}].")
            except:
                print("Warning: option: {} is not found in [{}].")
        return dict[option]

# Get the enabled modules
# An enabled module has a value of 1; otherwise 0
getEnabledModules():
    enabledModules = []
    for moduleName in _loadedModules:
        moduleDescription = ConfigSectionMap("" + moduleName + ", description"[][moduleName.lower()])
        moduleEnabled = ConfigSectionMap("" + moduleName + ", enabled"[][moduleName.lower()])
        if moduleEnabled == 1:
            enabledModules.append(moduleDescription)

    return enabledModules
```
The main window presents the investigator with an easy-to-use interface where it is only necessary to select the drive to run the tool on, and confirm which modules are going to be used. This is linked with the fact that on-scene investigators do not necessarily have the technical knowledge to use complicated applications (Dickson, 2013), and most configurations are executed prior to running the application. The Windows banner was added as a confirmation that the tool successfully detected that it was being run on a Windows system. The next few screenshots (Figures 15, 16, 17, and 18) show the final script of the main window interface running up to the moment it is making the call for the other modules.
7.2.2 Files by Hash Value

As seen previously, once the user has gone through the main window, the modules are then loaded. One of those modules is the “Files by Hash Value” and has three components: the script itself, a configuration file and a text file containing a list of bad hash values (Figure 20).

![Figure 19 - Files by Hash Value Folder Structure](image1.jpg)

The first part of the code (Figure 21) sets up the libraries, user interface and reads the configuration file in order to get the hash function to use, the file extensions to check as well as the path for the known bad hashes to use, which should typically be stored in the “knownbadfiles” folder (Figure 22). The second part of the code (Figure 23) is the main part of the process, as it is used to go through the drive and, if the file extensions match, calculate the hash using the user selected algorithm, such as md5 or sha1, and compare it to the list of bad hashes. If a match is found, an alert is displayed to the screen and logged for later analysis. Other metrics such as the starting time, length of the whole process the number of
files processed and matches found are also displayed. During the whole execution of the module is also displayed a live count of the files that have been processed, which keeps the user informed that the tool is running correctly. Finally, Figure 23 shows a screenshot of how the alerts are generated.

---

Figure 20 - Files by Hash Value Code part 1

---

Figure 21 - Files by Hash Value Configuration File
# walking through directories and files on the drive
# files with the current extensions are hashed
# the hash is then compared with the bad ones
def hashWalk(path, hashes, extensions):
    results = []
    mount = '/'
    for root, dirs, files in os.walk(path):
        for name in files:
            count += 1
            path = os.path.join('c:\', files)
            if name in extensions:
                current_file = os.path.join(root, name)
                output = open(current_file, 'r') as FIN:
                    FNAME = hashlib.md5()
                    for line in FNAME:
                        FNAME.update(line.encode('utf-8'))
                        hash = FNAME.hexdigest()
                result.append(os.path.join(current_file, name))
                print '{' + FNAME.hexdigest() + '}'
                return count
        for name in files:
            path = os.path.join('c:\', files)
            current_file = os.path.join(root, name)
            output = open(current_file, 'r') as FIN:
                FNAME = hashlib.md5()
                for line in FNAME:
                    FNAME.update(line.encode('utf-8'))
                    hash = FNAME.hexdigest()
                result.append(os.path.join(current_file, name))
                print '{' + FNAME.hexdigest() + '}'
                return count
    return 0

start()
7.2.3 Encryption Detection

The Encryption Detection module developed for this project is the Encryption Detection module and has only two components: the script and a configuration file containing the threshold to check and the minimum file size (Figure 24).

Figure 24 - Encryption Detection Folder Structure

The two main functions here (Figure 25 and 26) are the ones used to go through the files on the system and check their size, and pick only the ones which are above the minimum size set in the configuration file (Figure 27); and the one used to directly check whether a selected file is encrypted. The function’s theory to check encryption on a file was given by Mike Dickson during an interview as seen earlier in the literature and technical review, and should detect container sized file such as the ones created by TrueCrypt and password-protected ZIP files. Finally, Figure 28 shows a screenshot of the generated alerts.

```python
def enc(file):
  count = 0
  cnt = collections.Counter()
  with open(file, "rb") as f:
    for char in f.read():
      count += 1
      if (count % 5) == 0:
        cnt[chr] += 1
  high = max(cnt.values())
  low = min(cnt.values())
  percentage = int((float(low) / float(high)) * 100)
  return percentage
```

Figure 25 - Encryption Detection Code part 1

```python
def encDetect(minSize, threshold, path):
  result = []
  count = 0
  counts = 0
  cnt = collections.Counter()
  for root, dirs, files in os.walk(path):
    for f in files:
      try:
        count += 1
        current_file = os.path.join(root, f)
        sys.stdout.write("\r(\r) files processed\r\n")
        sys.stdout.flush()
        if float(os.path.getsize(current_file)) / (1024 * 1024) >= float(minSize):
          if (count > threshold):
            print "\nSUSPECT FILE FOUND ->\n\nCurrent file = " + current_file + "\n"
            except Exception, e:
              count -= 1
              sys.stdout.write("Could not process\r\n")
              sys.stdout.flush()
              pass
          result = ['files']
          return 0, count
```

Figure 26 - Encryption Detection Code part 2
7.2.4 Text-Based Searching

The Text-Based Searching module is a multi-purpose module and has two components: the script and a configuration file (Figure 29). The sub-modules part of it, such as looking for keywords in file names or contents, are defined by the user in the configuration file (Figure 30). In the script are then added different sub-modules code which are different but serve a common purpose, which is to look for keywords in certain strings. Again here, the first part of the code sets the configuration file to use and how to read it (Figure 31). The second part of the code (Figure 32) gets the respective names of the sub-modules to load and compare to that, runs them. Again, once a file with the correct extension is found, the keywords are applied and if there is a match, an alert is shown and logged for later analysis. Finally, Figure 33 show a screenshot of the generated alerts.

```
# configuration file for Encryption Detection
[encryptionDetection]
# This section is use for possible use of encryption on the system
# The minimum file size for encryption containers in MB and Threshold
encSize=5
threshHold=75
```

Figure 27 - Encryption Detection Configuration File

![Encryption Detection Generated Alerts](image)

Figure 28 - Encryption Detection Generated Alerts

![Text-Based Searching Folder Structure](image)

Figure 29 - Text-Based Searching Folder Structure
Figure 30 - Text-Based Searching Configuration File

```python
#!/usr/bin/env python

# Description: <Text-Based Searching> module for the python triage tool
# Author: Charley CÉLICE <celicecharly@gmail.com>

import os, fnmatch
import time
import datetime
import sys
import ConfigParser
import subprocess

# make it neat
os.system('clear')
print ''
print " < Text-Based Searching Module >"
print " "
print " Loading sub-modules..."

# reading the configuration file
# sub-modules name have to be added
Config = ConfigParser.ConfigParser()
_loadedSubModules = ["filesByContents", "filesByName"]
Config.read("./cfg.ini")
Config.sections()

# main definition for reading a *.ini file
def ConfigSectionMap(section):
    dicti = {}
    options = Config.options(section)
    for option in options:
        try:
            dicti[option] = Config.get(section, option)
        except:
            dicti[option] = None
            found = False
            print("exception on %s!")
            dicti[option] = None
    return dicti
```

Figure 31 - Text-Based Searching Code part 1
# sub-module descriptions and their respective keys
list1, list2 = getSubModules()
dictionary = dict(zip(list1, list2))
for subh in dictionary:
    print subh + " (" + dictionary[subh] + ")" + "

# sub-module #1
# Files by Name Keywords
def find(keywords, path, extensions):
    result = []
    count = 0
    for root, dirs, files in os.walk(path):
        for name in files:
            COUNT += 1
            sys.stdout.write(" \r\n files processed":format("%5d % count)
            sys.stdout.flush()
            for ext in extensions:
                if name.endswith(ext):
                    for key in keywords:
                        if fnmatch.fnmatch(name, key):
                            result.append(os.path.join(root, name))

    print "\n SUSPECT FILE FOUND ->" + os.path.join(root, name) + "\n"
    return {'files':result, 'count':count}

# sub-module #2
# Files by Contents Keywords
# etc...

def Start():
    extension2 = []
    keywords2 = []
    for subh in dictionary:
        fileExtensions = ConfigSectionMap(dictionary[subh])['fileextensions']
        extensions = fileExtensions.split(',')
        for ext in extensions:
            extension2.append(ext)
            keywords = ConfigSectionMap(dictionary[subh])['keywords']
            keywords2 = keywords.split(',')
            for key in keywords2:
                keywords2.append("* * keys + "*"
                print "* * subh + * found... Loading..." + ext
                print "Extensions:" + ext + extensions2
                print "Keywords:" + ext
                print "* * print * Starting...
                startTime = datetime.datetime.now()
                path = "C:\\"
                print "\n Scan started at %s" % startTime
                result = find(keywords, path, extensions2)
                elapsed = datetime.datetime.now() - startTime
                print "\n Script finished in %s (elapsed" + ext
                print "\n Files:
                for files in result['files']:
                    print files
                    print "\n Number of files found: %d % len(result['files'])

# Number of files processed: %d % result['count']

# sub-module #3
# Files by Additional Criteria
# etc...

Start()

def stop():

Figure 32 - Text-Based Searching Code part 2

![Text-Based Searching Module]

Figure 33 - Text-Based Searching Generated Alerts
7.3 Experimental Environment Implementation

As seen in the scientific experiments section, a virtual machine running has to be set up and it was also discussed in the design section that it would be a good idea to run it from a cloud platform. For this, the Edinburgh Napier University cloud is used to create a 50 GB Windows 7 image, where snapshots can be taken, and files and programs installed. The CPU is an Intel Xeon ES-2640 at 2.50 GHz and 4 GB of RAM is available. A set of various digital corpora files are downloaded to fill the memory completely. A total of 98,086 files and 45,656 folders are present on the first snapshot, which will be used as a scenario of IIOC. This should be enough to run a dozen of experiments to gather interesting results to compare. As in this scenario all the three modules: encryption detection, text-based search, and files by hash value search will be used, some of the files’ MD5 hashes are added to the text file containing known bad hashes, other files are renamed to suit the keywords in the configuration files, and encrypted containers are created using the TrueCrypt program. Python also had to be directly installed on the machine, due to some issues that are discussed in the section below.

7.4 Issues during Implementation

While developing the scripts, no important issues were encountered apart of small scripting mistakes that were quickly overcome. Most of the issues arose during the experimental environment implementation as many files were to be implemented on a virtual environment, thus taking a long time to download and manage; On top of that, internet connection issues arose from time to time in the cloud. It was however a good idea to use the cloud platform, as a local setup would have created more performance and time problems. There is now, however, a good virtual setup that can be used to run the different experiments detailed in the previous scientific experiment section.

It is possible to run Python off a USB flash drive by using “Portable Python” (Zivkovic, 2014), or directly convert the scripts into different executable to call instead. However, problems occurred when trying to run the scripts from USB drive. This is why Python was directly installed on the test machine to be able to run the scripts properly.

7.5 Conclusions

The main scripts for the triage tool to be developed in this project were implemented successfully and are working properly after testing all steps separately and carefully as the Agile method allow to do so, by modifying pieces of code until each function works properly, using quick development and testing stages; no particular issues were encountered in this stage. Only problems during the experimental environment implementation occurred, as it was a problem in the beginning to obtain a proper virtual environment and long to implement the desired files that will be used for the experiments. The setup is now ready to be used in order to start the experiments and compare the results of the prototype tool on the digital corpora files.
8 Evaluation

8.1 Introduction

In this evaluation chapter, the developed prototype triage tool will be run firstly using a validation type testing, in order to know if it performs as expected and achieves what it is supposed to achieve in terms of functionality. Then, using the experimental setup and measuring the metrics discussed in the design section, results are shown in tables summarising the experiments and then analysed and compared with other work. After those results are discussed, one or more other triage tools can be run to compare their functionalities with the developed prototype tool to highlight the main differences and potential problems. Those experiments will be using two scenarios that have been designed and discussed earlier: suspected paedophile and fraud investigation.

8.2 Triage Tools Experiments

8.2.1 Validation Type Testing

This section validates the functionalities of the developed prototype tool to verify that it meets its original objectives. In order to do that, the three modules are taken separately and tested to simply check if they work against manually created files.

For the encryption detection module, TrueCrypt is used to create a 1 MB and 5 MB sized encrypted containers. It has been seen in the literature review that TrueCrypt is a very well-known and easy to use encryption program, and is also easily accessible, which makes it a valid tool to use to create those containers. Those files are then placed in a directory along with other normal files and folders, and the module is run against this directory. After the module has been run, the 5 MB file is picked up as potential evidence and an alert is displayed. The 1 MB is not picked up, as the size limit in the configuration file is set to 5 MB, and other non-encrypted files are not picked up within this directory, which gives a successful run for this module. It can then be fairly accepted that this implementation of the encryption detection module works, and meets its objectives.

Concerning the text-based search, various files are placed within a directory and given different names and extensions including jpeg, doc, or txt files. One file is renamed as “something1 bar.jpg” and another as “something2 foo.txt”. The keywords “something1” and “something2” are added to the list of keywords to retrieve within the configuration file of the module, the file extensions “jpg, avi, doc” are also added as filters, and are then run against the directory. After the module was executed, both files were successfully picked up by the prototype triage tool, and the rest of the files ignored because even if they match the extensions, they do not match the keywords. This shows that the text-based search module works properly.
Finally, for the files by hash value module, various files are added into a directory, and the MD5 hashes of two of those files are added to the list of known bad files in the configuration of the module, using the Python `hashlib` library. One of those two file is a jpg and the other, a doc document. The list of file extensions to be used within the configuration of the module include jpg and txt only. The module is then run against this directory, and successfully picks the one file which MD5 and extension match. The third module, files by hash value, has then been successfully validated as working.

The other application that will be compared with the prototype developed within this project is TriageIR. This tool has already been validated in another paper (Shiaeles et al., 2013), therefore this step will be skipped, and it will directly be run in the next experimentation section to compare the functionalities and results with the prototype.

### 8.3 Triage Scenario Experimentation

#### 8.3.1 Scenario 1: Suspected Paedophile

After successfully implementing the experimental environment as described in the previous chapter, a snapshot is saved in order to run the tool ten times using the exact same conditions, as explained in the scientific experiment section. The results of the researched metrics are then extracted and displayed in the following tables, and colours are added to highlight interesting differences between results:
Table 3 - Experiment Results Tables (IIOC)

Those results show multiple things: the prototype tool picks up the evidences implemented in the experimental environment without missing any, but the execution time and number of files processed sometimes change. It is also interesting to note that for the encryption detection module, a number of false positive alerts are picked up, mostly because of some large zip files contained in the digital corpora but also because of some Windows system files. One of the encrypted containers is also picked up twice during the process, thus displaying two alerts for the same file.

Looking at the last table, the maximum average for the number of files processed seems to be 98,083. The system however contains 98,086 files in total. It then seems that some files are skipped or ignored from time to time, but the majority of the system is still processed. Furthermore, the most flagrant results here is the time spent processing those files: 38 seconds for the text-based search in the first iteration and 13 minutes in the fifth iteration is quite a big difference. Same for the files by hash value module, where the lowest time is 2 minutes and 04 seconds, while the longest is 13 minutes. This may be due to the fact that the modules are running in parallel, and on rare occasions conflicts between them may arise, causing the whole tool to slow down. Therefore, instead of ignoring those very different results, the average was calculated ignoring the minimum and highest values for total execution time. The average time however is still quite low for all the modules except the encryption detection. This important module has an average time of 1 hour and 20 minutes, which is normal considering it has to read large files byte by byte, increase and reset counters, but still above the expected time of one hour (section 2.3). However, the encryption module alerts on the encrypted container much before the module has finished running. It may therefore be enough for an investigator to apply to hold the suspect even while the tool is running.
The following graphs highlight the averages difference of time between the moments when the first piece of known evidence is found compared to the total execution time for each module:

**Encryption Detection**

**Text-Based Search**

**Files by Hash**

Figure 34 - Total Execution Time and Time of First Discovered Evidence
It can be clearly seen that for each module, even though the total execution time can be long, such as over an hour for the encryption detection module, the first pieces of evidence are displayed much before that. This is possible because for every valid hit the tool performs, an alert is generated automatically to the user. This is important because it means the investigator does not have to wait until the tool has finished running before they can make decisions on whether a digital device contains suspicious evidence or not.

8.3.2 Comparison with TriageIR

In order to highlight the differences between the prototype tool and a currently used open-source triage application, the TriageIR tool, discussed in the literature and technical review along with COFEE, will be run on the same snapshot to show the main differences in functionality and time-span. The following screenshots show the possible configurations of the tool, and the ones that were selected for running the experiment, namely system information, network information, and disk information, as they seem the most appropriate to select in this case:
When hitting “Run”, the tool prompt to download the Windows Sysinternals Suite to continue. After a successful download, it took an average of thirty seconds for the tool to run with the selected settings, and to store the gathered evidence into text files, as seen in the technical review.

This shows that it is more about doing a profiling of the system rather than directly look for suspicious evidence. It is also no more than a wrapper of the Sysinternals Suite for Windows. As highlighted in a research paper comparing TriageIR and two other open-source tools (Shiaeles et al., 2013), this may be good for malware analysis or machine profiling, but this is not as useful as the prototype tool alerting as soon as possible on suspicious evidence contained on a machine. It gives feedback to the user only once the tool has finished running, and only then it is necessary to analyse the reported evidence to make the decisions. Obviously, TriageIR did not find any useful evidence in this case. In what concerns the output of the tool, the investigator would be presented with a multitude of text files, as shown above, and no tools to process them automatically.

This shows the difference between one of the open-source triage tools available and the developed prototype for this project. Rather than just collecting information on the system, the prototype looks for particular possible pieces of evidence or clues that could lead to further investigation. It is however good to note that the TriageIR tool also possesses more advanced options, such as an automatic registry analysis, which could be rather useful to implement in a more advanced version of the prototype.

8.3.3 Scenario 2: Fraud Investigation

This scenario is mostly to check whether if running a module alone has a big impact on the results of the experiments. The text-based search module is directly run in the command line and the results displayed in the following table:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Number of Files Processed</th>
<th>Number of Generated Alerts</th>
<th>False Positives</th>
<th>Missed Evidence</th>
<th>Total Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98078</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:00:12</td>
</tr>
<tr>
<td>2</td>
<td>98077</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:13</td>
</tr>
<tr>
<td>3</td>
<td>98080</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:14</td>
</tr>
<tr>
<td>4</td>
<td>98224</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:25</td>
</tr>
<tr>
<td>5</td>
<td>98079</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:11</td>
</tr>
<tr>
<td>6</td>
<td>98083</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:09</td>
</tr>
<tr>
<td>7</td>
<td>98084</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:11</td>
</tr>
<tr>
<td>8</td>
<td>98079</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:07</td>
</tr>
<tr>
<td>9</td>
<td>98075</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:08</td>
</tr>
<tr>
<td>10</td>
<td>98080</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00:01:08</td>
</tr>
</tbody>
</table>

Table 4 - Experiment Results (Fraud)

Three points can be noticed from this table. The first one is that the average time execution for the text-based module run alone is 60.24% quicker than when run in parallel with the two other modules. The second point is that during iteration number four, it seems that around 200 more files were picked up in the process. Finally, the last point that can be noticed is that the first iteration took 19 seconds to complete, while the other ones took more than a minute, using the same exact snapshots. From this can be concluded that running multiple modules in parallel can have an impact on the performance of the prototype triage tool.
8.4 Conclusions

In this chapter, an evaluation of the implementation of the prototype tool was performed and its functionalities compared with another open-source triage tool. The experimental environment used for each experiment was carefully checked so the conditions are exactly the same for each iteration. This is important as this means it can be reproduced if someone else tries to run the same experiments and build on the work done during this project.

The prototype tool functionalities have also been validated using simple tests before running experiments to evaluate them. This was to ensure that it achieves what it is supposed to before running it on a large amount of files. The metrics used for the evaluation were the number of hits, meaning the number of generated alerts; accuracy, in other words if there were any false positives or missed evidence; and total execution time. It is however important to say that those scenarios and environment are not perfect conditions to evaluate such a tool. They were created to suit the tool’s current functionalities, which can be taken much further. For example, it does not check whether a jpeg file is really a picture and not a pdf. But it is good to show that such approach works in terms of giving a binary result on whether evidence might be present on a machine before doing a full disk analysis. If those scenarios were real, the investigators could safely assume that this machine is suspicious and should be more fully analysed.

It is also worth mentioning that, even though the evaluation and comparison of the prototype with another tool showed that the prototype’s forensic functionality is more useful in helping the investigator concerning whether it is necessary to seize a machine or not, more of the investigative administration functionality from the tools reviewed in section 3.3.3, namely COFEE and TriageIR, could also be added to the prototype. For example, it could be useful to add the possibility for the users to add case notes, including case ID and description, or even to save different configurations and selected modules under customised profile names. Also, the current prototype uses a console-based interface, and it might be a good idea in the future to investigate whether it would be a better approach to use a GUI. The issues to consider with this approach could include that it may help less technical users navigating around the tool, but perhaps it could also alter the performances as modules are added, because it would require more resources than a simple console application. This would be interesting to research the area of interface design and in particular forensic tools.
9 Conclusions

9.1 Overall Conclusions

Through the literature review, this project showed that new approaches in rendering Digital Forensics processes quicker are required. It has been seen that many challenges towards a solution have to be faced, such as an increased complexity of the tools to be developed, interfaces commonality, higher number of linked devices (S. L. Garfinkel, 2010), the difficulty to implement RAM analysis in applications (Yadav, 2011), issues with data encryption (Casey et al., 2011), the complex procedure involved in cloud forensics (Daryabar et al., 2013), and of course the exponential growing of memory in electric devices (Roussev et al., 2013).

Even though today’s most advanced forensic applications, used by law enforcement currently, are very reliable, they do not always assist the investigator in determining at the crime scene if evidence is present on digital devices. This can lead to large numbers of devices being seized and spending long periods of time performing analysis on the media and other associated devices (Daryabar et al., 2013; Dickson, 2013; S. L. Garfinkel, 2010). The solution proposed would drastically decrease the time spent doing full analysis that may in the end produce no useful results, simply by determining first whether such a time consuming analysis is required. This would help to solve some of the challenges faced in Digital Forensics. To implement this solution, a prototype design tool, experimental environment and experiments were designed to evaluate if the new tool would be successful in displaying, as it runs, useful alerts to the investigator, showing whether a digital device contains enough clues that may allow for further analysis. Through a technical review, it was also concluded that the target operating system should be Windows (NetMarketShare, 2013; W3SCHOOL, 2013), discussed how to improve current triage methods (Roussev & Quates, 2012), and that Python could be a good language to use for the prototype tool.

Finally, the design for the prototype triage tool was created. The requirements were drawn out based on the issues highlighted in the research phase, the experimental design was also looked into, as well as research into the associated scientific methods themselves. This helped discover useful metrics to use for gathering results from the experiments. Each module was designed separately and the design for the experimental scenarios and environment as well. The implementation of the tool as well as the implementation of the experimental environment was completed successfully even though issues arose due to technical reasons. An evaluation was then fulfilled in two steps for the prototype tool: validation type testing was carried out in order to verify the tool’s functionality, and triage scenario-based experimentations were then performed to evaluate the final prototype, and comparisons were made with another open-source triage tool. This helped to compare the functionality and usefulness, as well as performance and possible improvements of typical triage tools against the prototype. It has been found that the prototype tool is reliable in solving the initial challenges but that the experiments could be run in more scientific conditions and the tool could be further developed to meet the needs of forensics practitioners.
9.2 Aims and Objectives

9.2.1 Objective 1: Critically review the different approaches and techniques used in Digital Forensics

This objective has been successfully completed in the literature and technical reviews. In the literature review, the typical digital forensic approach was researched to have a better understanding of the processes to follow during an investigation, including evidence collection, preservation, analysis, and presentation (Carrier, 2002; Forte, 2003; S. Garfinkel, 2012; Justice, 2004; Kornblum, 2002; Kruse & Heiser, 2002); the challenges to overcome were then looked into in order to have a clear focus on what the problems are, such as the increased storage sizes and full disk encryption (Casey et al., 2011; Yadav, 2011); real world requirements were possible to look at because of the exceptional opportunity to interview the correct people in the field, namely practitioners from the Scottish Crime and Drug Enforcement Agency (SCDEA), providing with the conditions of a raid, the training they receive for using forensic tools, and examples of current triage tools (Dickson, 2013). The interviews with the practitioners were useful to confirm some of the research carried out in the rest of the literature review, namely a triage tool that would help the investigator to identify whether evidence is present on a digital device or not, and to design and implement the prototype tool; Traditional and triage digital forensic were then compared and different tools looked into to identify the suitability of those tools to overcome the current challenges; legality was also researched, including the ACPO guidelines as it is important when implementing a forensic application (ACPO, 2013; Haggerty & Taylor, 2006; Leyden, 2013; Sinangin, 2002; Vitak et al., 2011); finally Garfinkel’s digital corpora was evaluated, and was deemed useful in terms of building a scientific experimental environment for the experiments to be run with (S. Garfinkel et al., 2009).

In the technical review, the different possible platforms to focus on where researched, and a quick comparison of them in terms of accessibility, ease of development, popularity and ease of setup, and the differences between 64-bit and 32-bit computing was taken into account (Microsoft, 2013; NetMarketShare, 2013; W3SCHOOL, 2013); A deeper research into the role played by the Windows operating system in relation to digital forensics was conducted, such as an increase of complexity over the years (S. L. Garfinkel, 2010), including the current triage methods and how to improve them, mainly by looking at comparisons of open-source triage tools and a set of utilities provided by Microsoft which is commonly used (Microsoft, 2013; Russinovich, 2011; Shiaeles et al., 2013), for example by adding functionality to check if evidence is present on digital devices (Dickson, 2010; Roussev & Quates, 2012). The topic of Anti-forensics was looked into, and the scope for this project was also set, and it was decided that only encryption would be considered, and the potential access of a password-protected Windows machine, which helped determining which language could be used for the development of the tool. Finally, the digital forensics XML format, language choice and method of development were also evaluated, and it was decided that Python could be a good prototyping language to use.
All this review was very helpful for designing the prototype, but certainly could have been taken to a deeper level as the subject is large and complex. Some details were not considered due to time factors. For example, anti-forensic methods were very interesting and relevant, but it was decided not to include them in the scope of this dissertation and the current prototype as this may have caused the project to overrun. Also, it was planned to perform a much deeper comparison of the current triage tools used in the field, but it was not always possible as some of the tools currently used by law enforcement are not available to the public. Perhaps more interviews may have been a way of gaining more insight into these tools and their good and bad points. The literature and technical review were written as narratively as possible, in a way that information was presented in a logical way in order to get to the proposition of a solution to the initial challenges. It was challenging but a lot of information was found and new knowledge acquired; it was however, difficult to write well and at depth on such an interesting and wide subject within limited time. Many details thought to be important at the start of the project were taken out, for example in the general forensic approach, where only the important points useful for the rest of the dissertation were kept. Also, Mendeley has been very helpful in managing the resources researched.

9.2.2 Objective 2: Design an application based on the research and the review of other tools, and design a range of experiments for the evaluation

This objective has been successfully completed in the design chapter, where the requirements were drawn out, namely the main functionalities of the tool (Dickson, 2010; Roussev & Quates, 2012). Research into the scientific method, including the steps required for running successful experiments (Dodig-Crnkovic, 2002; S. Garfinkel et al., 2009; Mocas, 2004), and successfully evaluate the prototype tool (Beckett & Slay, 2007; Clayton, 2012), was also performed; the metrics to use were also researched, and it was decided that the ones to use could be the overall time it took the application to complete its operation, the hits or alerts generated, and whether these alerts are correct, or accuracy (Beckett & Slay, 2007; Clayton, 2012). Then, three initial modules were designed based on the literature and technical review carried out, namely the encryption detection, text-based search and files by hash value search. This was decided based on research conducted in the field showing that finding files by hash values is a standard way of identifying known pieces of data (Roussev, 2009), that encryption detection would be the only anti-forensic method part of the scope in this project (section 3.3.5), and that text-based search can be used in many situations, from searching files by name or contents to recovering internet history logs (Dickson, 2010; Ramsay, 2014); finally, scenarios were designed based on real-world scenarios and the experimental environment designed as well, including details on which digital corpora file types to use and how the snapshots are organised.

A fair amount of time was spent on the design of the prototype triage tool and it must be said that the outcome was quite satisfying, in terms of designing and implementing a novel prototype tool based on research, and considering that it was unknown, in the beginning of the project, whether such a prototype could be developed. Although it was difficult to design interesting experimental scenarios as well as the testing environment, the outcome here seemed to work well and produced interesting and useful results. Some difficulty came from
the fact that everything was new and many challenges occurred both from the topics researched and the technical aspects. Also, it was planned for some time near the beginning of the project to design a wrapper around the Sysinternals Suite utilities for the main part of the prototype, but this changed in a large way. This idea was not carried out through to the testing, because the TriageIR tool showed this was not an ideal way of doing things for triage. It did allow the tool to be compared with the prototype which also highlighted the advantages of creating the tool manually. In the end, it was a good choice to make, because the prototype triage tool developed now seems to provide better functionality in terms of what is useful to practitioners in the field.

9.2.3 Objective 3: Implement the application and run it against a range of experiments

This objective has been successfully completed in implementation section, where the prototype tool was implemented in Python using the Agile development methodology. This method of development, combined with an easy but powerful language was a good choice for developing the prototype and helped in resolving coding issues quickly and better implementations of the modules simply. The experimental environment was also implemented using Edinburgh Napier University’s cloud, as it has been seen in the design section that there are more advantages in running the experiments in a virtual environment running in a cloud infrastructure, such as the possibility to run the experiments in parallel (Apostu et al., 2013). Some technical issues during the implementation were encountered, including some issues with virtual machines and their networking that would prevent the downloading of digital corpora files, but these were overcome without too much delay.

Even if the prototype tool and experimental environment were successfully implemented, a number of issues appeared. During the implementation of the tool, the issues mostly concerned scripting errors, which were quickly overcome, and technical issues when using a USB device, which required Python to be directly installed on the machine for running the scripts. The scope of this project allowed for implementing a prototype, and a full implementation of the design would require more time. Concerning the experimental environment’s implementation, the conditions were not perfect. Even though the cloud infrastructure was a very good opportunity to run the experiments with a lot of resource and in parallel, compared to the other options of standalone machines. There were some network connection issues quite often for a period, thus not having an internet connection at all time, making the downloading of digital corpora files or other useful applications difficult at times. However, those issues were fortunately overcome in good time to perform the experiments and results still gathered in line with the project plan, which can be seen in Appendix 7.

9.2.4 Objective 4: Evaluate the results of the implementation of the final prototype

This objective has been successfully completed in the evaluation section, where the prototype tool was first validated in a validation type testing, to first check on the tool’s functionalities, showing the successful implementation; from that, the experiments were run on the selected
scenarios and the results gathered, compared, and discussed. It has been seen that even though the total execution time was higher than expected, the generated alerts still appeared in the early running time of the tool, showing that they could still allow an investigator to make decisions on whether taking a different course of action; the prototype’s functionalities were also compared with another open-source tool to spot the main differences. The prototype showed that the designed triage solution was more successful in evaluating whether the machine contained suspicious evidence, but that the TriageIR or COFEE tools also have more interesting functionality, such as the ability to analyse the Windows registry or create customised case profiles. This would be a good modular addition to the developed tool, for example if it is necessary to retrieve encryption keys (Carvey, 2011; Dolan-Gavitt, 2008) that could be useful to decrypt evidence in scenarios such as IIOC, or even look for suspicious registry entries.

9.3 Future Work

The biggest piece of work that could be taken forward from this project concerns the implementation of the prototype. More advanced users could implement this tool using much improved techniques that could speed up the processes and also add more modules. More functionalities in the current modules could also be added, as for example the current ones check for extensions in the filenames and not the magic numbers. It may also be possible to implement the prototype using other programming languages that may be more suitable for the task, such as C# or C++. Also, if more time would be available, a better design and implementation for the experiments could be performed to produce more data, and including the addition of more scenarios. It could also be interesting to look into a better visualisation of the gathered alerts and profiling of the suspected machine for example by using a GUI, to have a very clear and formatted feedback given to the investigator but also the analyst if needed, such as real-time graphs or other graphical means. It could also be possible to directly communicate the results to a remote investigator with more experience, in order to help make decisions on whether it is necessary to seize the digital media, or directly interact with the tool.

Other tests could also be carried out by law enforcement personnel in order to have feedback on whether this meets their expectations or if the current prototype could be modified to meet other requirements. It could also be possible to perform a better comparison of the prototype and other current triage tools that could not be accessed during this project via SCDEA.

The work conducted by Philip Penrose, mentioned in this project, concerning a sector sampling triage technique, could possibly be added to the prototype in order to test it against the realistic scenarios, and to test the modularity of the prototype to assess its flexibility for additional components.
10 References


Charley Célice – BEng (Hons) Computer Security and Forensics


Appendix 1  Initial Project Overview

Initial Project Overview
Title: An Application for Live Forensic Investigations on Windows Operating Systems

Overview of Project Content and Milestones
The focus of this project will be to develop an application capable of retrieving key evidence present on a computer running a windows operating system that could be used to help towards a live forensic investigation. The tool would help the user visualise the important data present on Windows, as quickly as possible. The content of the project would include:

- To research and understand the current tools and techniques used in today’s live forensic investigations
- To research the kind of data that would prove to be useful depending on a forensic investigation scenario
- To research and understand the legal aspects of such methods, mostly in a corporate environment
- To research the strength and the weaknesses of such existing applications
- To design the application taking in consideration all the previous points researched, create the application, and design a range of experiments for the evaluation
- If possible design a way to visualise in a more friendly way the acquired data such as using charts and graphs, which could lead towards a research on data visualisation in digital forensics for a Master’s degree
- Compare the test results with the expected results and other work in the area, and redact a general conclusion of the final product

The project milestones are as follows:

- Project Planning
- Literature/Technical Research and Review

C. Célise – Computer Security & Forensics – BEng (Hons)
Design and Implement the Application
Evaluation and Testing of Project
Redaction of the Project dissertation

The Main Deliverable(s)
A well-designed and ready-to-use application, possibly embedded on a USB pen drive requiring no – or a very fast – setup and a proper documentation.

The whole research and development process will be detailed properly in the form of a final dissertation report.

The Target Audience for the Deliverable(s)
Principally small and medium sized organisations, possibly individuals, willing to perform their own forensic investigations on suspected machines are targeted. More particularly the companies with strict corporate rules regarding what a person can and cannot do with one of their machines, within the corporation.

The Work to be Undertaken
This project will involve several steps, beginning with a research around the subject of digital forensics and narrowing down to the current techniques used for today’s live forensic investigations within an organisation. Interviewing a person working in the force could be a good opportunity to support the research and develop other ideas.

After what, an application targeted specifically for the purpose of investigating a suspected employee within a business will be designed, developed and tested. Such a tool could also be tested against different antivirus software, and be trustful.

Additional Information / Knowledge Required
Acquire in-depth knowledge of the methods and the legal aspect in digital forensic investigations in general and within a business.

C. Célise – Computer Security & Forensics – BEng (Hons)
Greatly develop skills in the C# programming language and in Python, as well as other programming and scripting languages if the need arises.

Understand the data acquired by the tool and put it in the context of an investigation, to be able to use it as inculpatory or exculpatory evidence.

Start building knowledge towards data visualisation in digital forensics.

**Information Sources that Provide a Context for the Project**

Journals, articles and papers – mostly peer-reviewed and conference papers – around the topic of forensic applications and digital forensics in general will be the most used. Here are some examples of such papers:


Other sources of information from various websites and books will also help learning and understanding what this project is aiming at.

The interview of a person working in the force could help greatly in the making of this project.

**The Importance of the Project**

Computer misuse or the break of corporate rules by employees within a business is more and more common. This type of fraud is usually dealt by the forensic force, but the need of
tools that can be used by the corporate owners themselves – via for example the I.T. branch – is rising.

This is where the importance of the project is: having available tools meeting the forensic standards and recognised by the jurisdiction, if possible free as most of the currently available tools are very high-priced, and made available to companies and individuals.

**The Key Challenge(s) to be Overcome**

The main challenge will be to develop an application that meet the current standards and is recognised as usable, ethically and juristically speaking, taking into consideration the research done in the field.

The technical challenge of designing and coding the application is also involved.

Another challenge will be to decide which information – both researched and discovered by using the tool – is important, and to understand and interpret it.

Last but not least, acquiring in-depth knowledge in the legal aspect of the project will be a key challenge to be overcome.
12 Appendix 3 Some Diary Sheets

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Charley Célisce
Date: 19/09/2013

Supervisor: Richard Iacovartane
Last diary date: 10/09/2013

Objectives:
- Check Richard's email on things to look at
- Research methods
- Management
- Report
- Start researching around this subject
- Start IPO draft
- Start a Project plan
- Start preparing diary sheets
- Think of a title to put on the Project Registration Form and submit the form

Progress:
- Gathered a few papers on the subject and added to the reference database
- Project plan draft completed
- Diary sheets prepared
- Dropbox account setup / files added
- Reviewed some past honours project reports

Supervisor's Comments:
Excellent progress so far. Well under way and excellent idea for a prototype tool. Continue to research tools, seek out academic and develop IPO for next week.

Version 2
Edinburgh Napier University
SCHOOL OF COMPUTING

PROJECT DIARY

Student: Charley Célise

Date: 02/10/2013

Objectives:
- Continue literature research notes
- Start chapters into literature review

Progress:
- Started Abstract draft
- Project overview / background / aim and objectives
- Dissertation structure draft / literature review plan
- Done an excel document to gather literature research

Supervisor’s Comments:
Great start to research db and documents. Continue with research and talk to SCADA chiefs covering investigation/tools they might use/want to use.

Version 2

Edinburgh Napier University
SCHOOL OF COMPUTING

PROJECT DIARY

Student: Charley Célice
Date: 31/10/2013

Objectives:
- Continue research on direction of a triple type implementation and add to report
- Try to plan a meeting with Mike Dickson from the GDEA

Progress:
- Re-work most of literature review/technical review
- Did some more research on triple
- Meeting with Mike Dickson done
- Did some Python testing

Supervisor’s Comments:

Excellent progress in report, very well prepared for trip meeting.
Continue with research/it review, and possibly add context page/let’s for meeting next week.

Version 2
Edinburgh Napier University
EDINBURGH NAPIER UNIVERSITY
SCHOOL OF COMPUTING

PROJECT DIARY

Student: Charley Céllice  Supervisor: Richard Macfarlane
Date: 03/12/2014  Last diary date: 27/01/2014

Objectives:

- Review module handbook – report parts marking
- Design sections started with schematic and refs to other research

Progress:

- Enhanced literature and technical reviews, and added a few things
- Properly started writing on the hinge tool design, and structured the plan better
  based on an MSc you gave me to look at.
- Schematic of the design

Supervisor’s Comments:

Good work on design so far. Higher level schematic with connectors A’s incorporated. Bring
forward research from earlier hinge
design.

Version 2

Edinburgh Napier University
EDINBURGH NAPIER UNIVERSITY
SCHOOL OF COMPUTING
PROJECT DIARY

Student: Charley Célice
Supervisor: Richard Macartain

Date: 7/3/2013
Last diary date: 17/03/14

Objectives:
- Continue design section
- Add experimental design/ reasoning
- Expand on diagrams for design
- Add something on digital corpora as part of experimental design

Progress:
- Added new diagrams
- Modified chapter
- Modified font size: is it better?

Supervisor's Comments:
Good progress with report. Design needs some more licence to research and technical review. Start experimental setup as soon as possible.
13 Appendix 4 First Emails Exchange with Mike Dickson

From: Célice, Charley [mailto:10012794@live.raper.ac.uk]
Sent: 07 October 2013 13:33
To: Dickson, Michael
Cc: Macfarlane, Richard
Subject: Project Windows forensic tool

Dear sir,

I am the student in question doing the project on developing a forensic tool.

Let me start by saying this is a great opportunity to be in touch with you and am very glad!

To explain the project as precisely as I can, it will be a tool fairly easy to use, but to access only the Windows artefacts that one could reach even just with a mouse and a keyboard, such as the list of users, their last login time, web browsing history, temp folder, some network statistics, a history of external devices, and so on! This is definitely not EnCase, but I believe it is a good project for me to get started in the area.

The target audience I chose are companies and maybe individuals if they want to do their own "low level" kind of forensic investigations on their machines. After all, if they can find some evidence with an easy tool by themselves at first, why do they need to invest so much time and effort doing a full forensic investigation by external professionals?

I hope this background is enough for you so far! I would be very glad with any input you would have that could help me get this project done, any experience you may have had in the area, even some advice when building my tool that could make it as much reliable and efficient as possible... or even have a look at the prototype when it’s finished?

Any input at this stage would be very useful. Also, I hope you have no problem with me using your name in my dissertation if needed?

Regards,

Charley.

From: Dickson, Michael
Sent: 7th October 2013 19:56
To: Célice, Charley

Hi Charley –

It sounds like an ideal project. Perhaps you ought to start off by deciding upon exactly what purpose it will serve and then deciding which features it will have (ie, what it will report on) that will support that target audience.

Have you seen Microsoft COFFEE? It seems to do a few things that your project will cover.

I did once design a binge tool for use by law enforcement, but that sounds like it may be a little different in scope to what you have in mind. Are you thinking that your tool will run in a matter of a few seconds, as opposed to scanning the system for an extended period?

I’d be very happy to look at any prototypes you have!

Kind regards

Mike

The information transmitted in this email is intended only for the person or entity to which it is addressed and may contain material and/or privileged information. Any communication, disclosure or action in reliance upon the information by persons or entities other than the intended recipient is prohibited. If you received this in error please contact the sender and delete the material from any computer.

NOT PROTECTIVELY MARKED

Charley Célice – BEng (Hons) Computer Security and Forensics 80
From: Célice, Charley [mailto:10012794@liv.ac.uk]
Sent: 08 October 2013 16:06
To: Dickson Michael
Cc: Mezfarlenes, Richard
Subject: Re: Project Windows forensic tool [NOT PROTECTIVELY MARKED]

Hello,

Thanks for answering!

Well to answer your question, yes I have had a quick look at Microsoft COFFE, however it seems it is only legally available to the enforcement law.

Concerning the exact purpose of the tool, I have made a couple screenshots of a very early version of the prototype that I have started some time during the summer.

I have hopes that the tool will run in a matter of a few seconds instead, but I may add an option to run the most important tools all at once and get a report out of it, which may take a bit longer. Again this is just an idea!

Thanks Mike!

Regards,

Charley,

From: Dickson Michael
Sent: ?Wednesday?, 7/07/2013 7:07 PM
To: Célice, Charley

NOT PROTECTIVELY MARKED

Hi Charley –

That looks outstanding so far.

I didn’t realise COFFE was only available to law enforcement. I suppose I take that for granted here. What it really is, is just a collection of Microsoft command line tools under a single ‘wrapper’ program which directs their output. It has the same sort of functionality as you are after, but is a bit messy.

On that subject – how much impact does running your program have on the system it is examining? What sort of ‘forensic footprint’ does it leave behind? Can it be used unobtrusively?

One suggestion I have, which may impact on the running of the program a little – how about encryption detection?

Kind regards

Mike

NOT PROTECTIVELY MARKED

From: Célice, Charley [mailto:10012794@liv.ac.uk]
Sent: 09 October 2013 17:27
To: Dickson Michael
Subject: Re: Project Windows forensic tool [NOT PROTECTIVELY MARKED]

Hi,

I will have to find some methods to know if the program leaves some traces, however at first glance, it doesn’t seem to be the case. But it is definitely in my scope!

Encryption detection was in my mind, but it can be quite challenging, I will try to find ways...

It is also likely that if I cannot do something for a particular reason, I will have my application call another tool. For example, design a way to call EnCase for one particular task (this is just an example!).

Do you think a company would have a good use of such a tool? Do you think persons like you would have a use for it?

I am very glad you liked the idea so far! I am doing the best I can, but there is still much to do.

Regards,

Charley.
From: Dickson Michael
Sent: Thursday, 10 October 2013 13:25
To: Charley, Charley

NOT PROTECTIVELY MARKED

Hi Charley –

The easiest way to detect a program's footprint is to run a registry monitoring tool to determine what registry changes it makes when it is run.

If your tool doesn't parse the computer's filing system in its own then it will very likely update many artefacts as it traverses the disk.

Encryption detection isn't that hard. There are a load of ways of doing it, but the easiest is to consider that in encrypted data there will be an even spread of instances of all 256 byte values

Calling EnCase seems a bit too heavy. That would be like me writing a word processor by writing an EXE that called Word :)

Small companies may well have a use for a tool like this – larger companies will likely have their own solutions.

People like me wouldn't. What I would be after is a triage tool that will determine whether or not a computer would have data likely to be of interest to me in the first place. If you want my triage design then I can send it to you. The main difference is that triage tools have to assume that the end user is monumentally stupid. :)

If there is anything I can do to help, let me know. If you want to meet up that's OK too.

Kind regards

Mike

NOT PROTECTIVELY MARKED
Appendix 5  Planning Meeting with Mike Dickson

From: Celice, Charley [mailto:10012794@live.napier.ac.uk]
Sent: 21 October 2013 13:38
To: Dickson Michael
Cc: Macfarlane, Richard
Subject: Meeting?

Hello Mike,

Have you seen my previous email?

I have a meeting every week with Richard, I was wondering if you would be available some time this week to meet with us? Maybe at the university or anywhere else where it is convenient enough for you?

I understand that you may be busy! Please tell me if you have any time/place suggestions if it is possible to have a small meeting.

Regards,
Charley.

The information in this e-mail and any attachment(s) is private and may also be LEGALLY PRIVILEGED. It is intended only for the addressee. If you are not the intended recipient or responsible for delivering it to the intended recipient, you are hereby notified that any use, disclosure, review, dissemination, distribution or reproduction of this e-mail is strictly prohibited.

If you have received this e-mail in error please advise the sender immediately by using the reply facility in your e-mail software or contact POLICE SCOTLAND on 0800 786 289070 and delete the e-mail.

From: Dickson Michael
Sent: 21 October 2013 16:13
To: Celice, Charley
Cc: Macfarlane, Richard

Hi Charley –

I didn’t see it, sorry

When would suit you? It will have to be this week as I am away in Coventry all next week teaching there.

Meeting at Napier would be fine.

Mike

From: Celice, Charley [mailto:10012794@live.napier.ac.uk]
Sent: 21 October 2013 15:13
To: Dickson Michael
Cc: Macfarlane, Richard
Subject: Re: Meeting? [NOT PROTECTIVELY MARKED]

Hi,

Do you have any free spots on Thursday after 12 noon? For example at 1pm in Richard’s office?

Regards,
Charley.

From: Dickson Michael
Sent: 22/10/2013 08:09
To: Celice, Charley
Cc: Macfarlane, Richard
Subject: RE: Meeting? [NOT PROTECTIVELY MARKED]

Hi Charley –

It will have to be after 3:30pm some day this week.

Mike

From: Celice, Charley [mailto:10012794@live.napier.ac.uk]
Sent: 21 October 2013 13:38
To: Dickson Michael
Cc: Macfarlane, Richard
Subject: Meeting? [NOT PROTECTIVELY MARKED]
From: Celice, Charley [mailto:10012794@live.napier.ac.uk]
Sent: 22 October 2013 13:44
To: Dickson Michael
Cc: Macfarlane, Richard
Subject: RE: Meeting? [NOT PROTECTIVELY MARKED]

What do you think about 4pm on Thu Mike?

Regards,

Charley.

From: Dickson Michael [Michael.Dickson@scotland.run.police.uk]
Sent: 22 October 2013 13:48
To: Celice, Charley
Cc: Macfarlane, Richard
Subject: RE: Meeting? [NOT PROTECTIVELY MARKED]

NOT PROTECTIVELY MARKED

Yes, that will be fine. Richard's office? Remind me which one that is? (It has been a while)

Mike

NOT PROTECTIVELY MARKED

From: Macfarlane, Richard
Sent: Thursday, 24 October 2013 7:17:54
To: Dickson Michael, Celice, Charley

Hey chaps,

We can grab a coffee first if you like. I'm on 0777 254 3394, or give Charley a call when you arrive and we can swing down and meet you at reception.

Rich

From: Celice, Charley [mailto:10012794@live.napier.ac.uk]
Sent: 22 October 2013 15:38
To: Dickson Michael, Macfarlane, Richard
Subject: RE: Meeting? [NOT PROTECTIVELY MARKED]

This sounds good. Reception at 4pm Thu?

Regards,

Charley.
Appendix 6

Philip Penrose Quick Crypto Identifier

Celice, Charley

to Penrose, Philip
cc Macfarlane, Richard

Re: Quick crypto identifier

Hello Philip,

I am the honours student in question...

The basic method from Mike Dickson is as follows:

1. Read every 5th byte in the file and increment a counter (one of 256) for that byte value
2. Continue to the end of the file
3. Take the lowest of the 256 counters and express this as a percentage of the highest counter
4. If the percentage is over a given threshold (say 75%) then the chances are the file is encrypted

I have tried to follow this theory and implement it in Python... I have had some good results...
I would be curious to know what kind of files I could test this against? I have tried with TrueCrypt containers (1mb and 2mb) / Encrypted zip files.

See attachment for the Python file.

Regards,

Charley

Penrose, Philip

to Celice, Charley
cc Macfarlane, Richard

RE: Quick crypto identifier

Hi Charley,

The algorithm that you are using is just checking if the file that you are looking at is reasonably ‘uniform’. It relies on the fact that encrypted files should not show any pattern and thus no particular character should occur more than any other, otherwise they would be susceptible to cryptanalysis.

If you were looking at an HTML file then the characters ‘<’ and ‘>’ would occur frequently but many characters (ASCII codes > 127) would not occur, so minimum count / maximum count will be close to zero. If characters are uniformly distributed then each character has a similar count and so min / max will be close to 1.

Thus it would seem on the face of it that your Quick Crypto Check (QCC) should work. However, this is only true for large files. With a large file (>1 MB say) the assumption of uniformity is more likely to be true and be increasingly so with larger files. My work is with small file fragments (e.g. one disc sector or about 4 KB). At this size the uniformity assumption is less likely.

I did a quick and dirty implementation of your algorithm in C# that will test all files in a folder. I’ll put in a link to a dropbox folder that contains a few hundred encrypted fragments that I used for testing. The QCC folder holds the C# project so that you can check it (I’ve attached the exe as a zip). I tried it first with 4 KB fragments and it didn’t detect any of them. I added 8 KB fragments and still none were detected. You can test these fragments with your Python code as well. I’ll have a look at it later today.

You can see from the minimum and maximum values shown when you run the program that no uniformity has ‘built up’ at this small scale and so these won’t be detected by your algorithm.

Regards

Phil
Re: Quick crypto identifier

This is understandable; Mike D. mentioned that this “Quick Algorithm” would check only files of size over, for example, 5MB. Now I see why.

Thank you.

Regards,

Charley.

From: Penrose, Philip
Sent: Monday, 4 November 2013 21:00
To: Célice, Charley
Cc: Macfarlane, Richard
Appendix 7

Project Plan