UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Considerations Towards the Development of a Forensic Evidence Management System

Submitted in partial fulfillment of the requirements for the degree

Magister Scientia (Computer Science)

in the

Faculty of Engineering, Built Environment, and Information Technology

at the

University of Pretoria

Kweku Kwakye Arthur

July 8, 2010

Abstract

The decentralized nature of the Internet forms its very foundation, yet it is this very nature that has opened networks and individual machines to a host of threats and attacks from malicious agents. Consequently, forensic specialists — tasked with the investigation of crimes commissioned through the use of computer systems, where evidence is digital in nature — are often unable to adequately reach convincing conclusions pertaining to their investigations.

Some of the challenges within reliable forensic investigations include the lack of a global view of the investigation landscape and the complexity and obfuscated nature of the digital world. A perpetual challenge within the evidence analysis process is the reliability and integrity associated with digital evidence, particularly from disparate sources. Given the ease with which digital evidence (such as metadata) can be created, altered, or destroyed, the integrity attributed to digital evidence is of paramount importance.

This dissertation focuses on the challenges relating to the integrity of digital evidence within reliable forensic investigations.

These challenges are addressed through the proposal of a model for the construction of a Forensic Evidence Management System (FEMS) to preserve the integrity of digital evidence within forensic investigations. The Biba Integrity Model is utilized to maintain the integrity of digital evidence within the FEMS. Casey's Certainty Scale is then employed as the integrity classification scheme for assigning integrity labels to digital evidence within the system.

The FEMS model consists of a client layer, a logic layer and a data layer, with eight system components distributed amongst these layers. In addition to describing

the FEMS system components, a finite state automata is utilized to describe the system component interactions. In so doing, we reason about the FEMS's behaviour and demonstrate how rules within the FEMS can be developed to recognize and profile various cyber crimes. Furthermore, we design fundamental algorithms for processing of information by the FEMS's core system components; this provides further insight into the system component interdependencies and the input and output parameters for the system transitions and decision-points influencing the value of inferences derived within the FEMS.

Lastly, the completeness of the FEMS is assessed by comparing the constructs and operation of the FEMS against the published work of Brian D Carrier. This approach provides a mechanism for critically analyzing the FEMS model, to identify similarities or impactful considerations within the solution approach, and more importantly, to identify shortcomings within the model. Ultimately, the greatest value in the FEMS is in its ability to serve as a decision support or enhancement system for digital forensic investigators.

Acknowledgements

In loving memory of Mrs Afua Kobi Andoh

My greatest appreciation goes to:

- God Almighty, for His endless favour, and for providing me strength and endurance to complete this work.
- Hemant "*Holomisto*" Grover for encouraging me to enrol for the Masters programme. It hasn't been easy, but you helped me make a great decision bhai!
- Professor M.S. Olivier for his supervision. I especially appreciated your thoroughness, your demeanor and leadership approach, uncanny ability to identify and describe the most discrete items, and your vast theoretical, technical, social and philosophical knowledge. A special word of thanks to Professor J.H.P. Eloff and Professor H.S. Venter for their contributions, especially during the paper-writing process.
- Past and present colleagues in the Information and Computer Security Architecture (ICSA) Research Group for their support and encouragement — Vafa Izadinia, Marco Slaviero, Heiko Tillwick, Thorsten Neumann, Neil Croft, Emmanuel "*Emax*" Adigun, Maciej Rossodowski, Francois Mouton, Abiodun Modupe, Johan Fourie, and Kamil "*Kamillionaire*" Reddy. A special word of thanks to Emmanuel and Maciej for introducing me to, and coaching me in the use of LaTeX

- My family, Mr Kweku "*Dada*" Arthur(Snr), Mrs Theresa "*Mumzo*" Arthur, Kojo "*Du*" Arthur, and Nana "*Nanzo*" Arthur.

- My cohorts and henchmen, Kwaku "*Chief*" Kyereh, Nkgomeleng "*Jigga*" Thobakgale, Tasha "*Bleek*" Chapeshamano, Bheki "*maBiza*" Khumalo, Sydney "*Syd*" Macauley, Angela "*Anj*" Parry-Hanson, and Nhlakanipho "*Ash*" Cebekhulu. Your encouragement, gentle urgings, and many late night outings saw me through a great deal.

- My past and present management at SARS, Dr. Hettie Booysen and Tony Apsey. Thank you for your input and support over the duration of my studies!

- The SARS IT Security Operations team. Special thanks to Lilly Lesejane, Werner van den Heever, and Tebogo Selamolela for their interest and encouragement.

- Jim Cracknell, my boet and "*da man*"! Thank you for being a sage, and for sharing the secret to life — education. I treasure all our interactions, the laughs, the 'searching' questions, and the many talks on life, theology, academia, corporate environments, and the enigma that is woman.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Modern society and the Internet

The Internet has proved to be an indispensable tool in modern society, and although intangible, this "commodity" has significant influence in many social, financial and technological contexts. For years it has enabled the sharing of resources between millions of homogeneous computer systems, and subsequently, heterogeneous computer systems around the world. It has become the backbone of on-line trading, "instantaneous" communications and also serves as a rich source of educational and entertainment material.

The existence of the Internet has largely endorsed the economic principle of perfect information; a principle in which *"people can acquire most of the information that is most relevant to their choices without great difficulty"* [49, 27]. Levitt and Dubner [49] describe information as *"the currency of the Internet"*. Furthermore, they attribute much of the Internet's success to the efficiency with which information is exchanged between experts and the public. Undoubtedly, the advent of the Internet has positively impacted all aspects of day-to-day life.

Fundamentally, the Internet is a world-wide computer network, interconnecting millions of computing devices. With these connections, Internet infrastructures are able to store and transmit protocol information for applications such as electronic mail, file transfer, Internet telephony, multimedia content, remote access (to network

devices) and World Wide Web content [46]. The decentralized nature of the Internet forms its very foundation, yet ironically, this very nature has opened networks and network hosts to information security threats and attacks from cyber criminals.

Using the appropriate computing devices, cyber criminals are able to seamlessly conduct malicious or criminal acts through the Internet [8, 75, 98, 28]. Examples of these crimes include cyber obscenity, financial fraud, credit-card misuse, or the theft of trade secrets. In its 2007 annual report [71], the Computer Security Institute highlights the prevalence of information security and computer-related incidents and illustrates the financial losses thereof. In essence, this survey demonstrates the nature and number of exploits that are increasingly carried out through the Internet's facilities.

In light of the increasing incidence of cyber crimes, a new investigative competency has developed, namely computer forensics. This field exists for the investigation of crimes commissioned through the use of computers, where evidence is electronic in nature [74, 17, 64]. However, with the prevalence of digital devices, this field has been broadened to include the investigation of all digital devices and is now commonly referred to as digital forensics. A computer forensic specialist (CFS) is therefore tasked to investigate a digital crime scene. The specialist impartially scrutinizes a number of digital sources (that are involved or thought to be involved in the crime) and ultimately produces documentation summarizing the contents of the investigated digital sources. Although this field is within its infancy [22], all investigative results are achieved with the use of forensic hardware devices, specialized forensic tools and stringent forensic processes and investigative methodologies.

Henceforth, the terms computer forensics and digital forensics, and computer forensic specialists and computer forensic investigators will be used interchangeably.

Naturally, the primary aim within a digital forensic investigation is the successful prosecution of cyber criminals. Therefore, similarly to other forensic sciences, computer forensics relates law and science. Alternatively stated, CFSs are tasked *"to find facts in the form of electronic evidence that can be presented in a coherent way so that others may weigh that evidence and then assign guilt or innocence where appropriate"* [98].

1.2 Challenges within digital forensics

The digital forensics fraternity is not without its challenges. However, in the author's opinion, the primary challenge facing this field is the fundamental difference between the physical world and the digital world.

The physical world represents a realm where intangible properties such as time, space, identity, or physical location can not be controlled or amended by agents (humans) within the environment. This world is also characterized by its deterministic and finite nature, where actions have a definitive source and destination.

On the other hand, the digital world is one where any single action is virtually independent of time and physical location, and more often than not, the source of an action is obfuscated. In the digital world, with the appropriate knowledge, properties such as time, identity and location can be amended.

This distinction is clearly important in the field of digital forensics, where it is often easy to prove something in the physical world, but difficult to attribute it to someone in the digital world. In subsequent paragraphs a brief discussion on the core challenges within this profession is provided.

The perpetual growth in technology and Internet services continues to introduce new attack vectors within the digital realm. In certain instances, new technology and services are used (or rather misused) in ways that were never considered. As a result, existing forensic tool-sets are unable to adequately examine digital evidence contained within some new technological devices. Burke and Craiger [9] provide an overview of forensic analysis of Xbox consoles; this undoubtedly illustrates the influence of technological growth on digital forensics. In addition, the investigation search area has "mushroomed" in wake of increased storage device capacities.

There is also a general lack of standardization within the forensics field; this sentiment is echoed within [55, 48]. Meyers and Rogers [55] highlight the requirement for standardization and certification within computer forensics. These requirements are expounded through Meyers and Rogers' examination of federal and state court cases (criminal and civil). Similarly, Leigland and Krings [48] call for the formalization of forensic investigative procedures. Based on mathematical foundations, Leigland

and Krings construct a forensic investigation framework that can be tailored to the investigation scenario at hand.

In many instances, the desired penalty for cyber crimes are largely inconsistent with the tangible (or intangible) impacts of such crimes. This is due to the lag between law and forensic sciences — a situation that potentially renders investigations worthless. The issue of legal jurisdiction is one that must also be considered within a digital forensics context. In reality, the physical location of the source and destination of a crime may be significantly different, for instance, Japan and England. Unfortunately, there is also disparity between the legal texts and penalties within differing countries — a situation which further antagonizes efforts to thwart cyber crime.

A subtle, yet significant caveat within all forensic investigations is the importance of evidential integrity. However, due to the 'volatility' of digital evidence [5], the aspect of evidential integrity is somewhat pronounced within a digital forensics context. In this context, the word 'volatility' is used to describe components of digital devices, such as random access memory, which may be analysed during an investigation but are generally known to be volatile in nature [81]. Furthermore, the word 'volatility' is broadly used to incorporate the ease with which digital evidence can be created, altered, or even destroyed during investigations [36]. It is upon this basis that due-diligence must be exercised during all digital forensic investigations to ensure legal admissibility of evidence.

The challenges facing the field of digital forensics are comprehensively outlined in the texts above. Although these challenges are significant, they will all not be considered within the scope of this research. Rather, the remainder of this dissertation will focus on the challenge of evidential integrity within digital forensic investigations. A key obligation of any digital forensic investigator is to protect and preserve the integrity of digital evidence; this is to ensure the admissibility of such evidence within disciplinary, civil, or criminal proceedings. To date there is arguably a marked deficiency in the body of knowledge addressing this particular aspect of digital investigations — many professional and academic texts are either dedicated to forensic processes, or to the tools commissioned within forensic investigations. It is for this reason that we propose a mechanism to maintain the integrity of digital evidence

involved within digital forensic investigations.

1.3 Problem statement

In this dissertation we propose and describe a framework for an integrity-aware forensic evidence management system (FEMS). In describing this framework, fundamental principles regarding data integrity, data classification and information flows are considered. The Biba Integrity Model [66] is employed to preserve and reason about the integrity of stored evidence. As a requisite to this integrity model, Casey's Certainty Scale [13] is utilized as an integrity classification scheme in assigning integrity classes to evidence within the system. Finite state automata theory is then employed in illustrating the system component interactions and the system behaviour thereof; at this stage the potential for achieving cyber crime profiling is explored. Specifications for the construction of the FEMS are derived from principles described by Taylor et al [87]. As such, properties of the proposed FEMS are:

- to manage digital evidence and the integrity thereof,
- to preserve the integrity of evidence within the system, and
- to provide a degree of automation within the analysis stage within forensic investigations.

Therefore, similarly to an expert system, the FEMS would utilize a knowledge base to provide insights into investigative hypotheses and inferences in a heuristic manner, based on rules within the system.

1.4 Dissertation layout

This dissertation consists of nine Chapters. The current chapter, **Chapter 1**, provides the reader with an introduction to the report by describing the nature and exploitation of the Internet by cyber criminals. Thereafter, challenges facing the field

of digital forensics are highlighted. Finally, the problem statement outlines the author's contribution toward aspects of evidential integrity and evidence management.

In **Chapter 2**, some depth into the issue of cyber crime and its impact is given. Some characteristics and motivations of cyber criminals is also provided in this chapter. A history of forensic sciences and an overview of computer forensics is provided in **Chapter 3**. The chapter further provides insight into the tools commissioned during computer forensic investigations.

Due to the proposed model's reliance on trace or log evidence, **Chapter 4** considers the current state of logging and log correlation; log files are vital instruments in the reconstruction of the activities leading to (and after) a cyber incident. For this reason, the fundamental concepts, techniques and challenges within logging and log correlation are examined in the remainder of the chapter.

Chapter 5 constitutes the core contribution of this research report, wherein an overview of the proposed forensic evidence management system (FEMS) is provided. In addition, we illustrate the overall system design and discuss each of the system components, highlighting their usefulness within the proposal.

In **Chapter 6** the FEMS component interactions are illustrated using finite state automata (FSA) theory. At this stage, the concept of cyber crime profiling is considered. Furthermore, it is within this chapter that the system's heuristic behaviour is described. This behaviour is derived from the evidence entered into the system and the active rules within the system.

Chapter 7 expands on the states and transitions of the FEMS FSA that is illustrated and described in Chapter 5. In particular, we make use of flowcharts to describe the inner-workings of the FEMS FSA states and transitions. Thereafter, we develop processing algorithms for the states within the FEMS FSA.

In **Chapter 8** we provide a critical assessment of the FEMS against the PhD work of Brian D Carrier. We commence by presenting the problem and solution approach within Carrier's work. Thereafter, we perform a comparison between our works, to identify similarities and differences between our proposed models and our solution approaches.

Chapter 9 summarizes the work presented in this dissertation and reflects on the

author's contribution. The chapter also provides a brief discussion on the limitations within this work, thereby directing the reader towards areas for further research.

A list of references consulted for this research is provided in the **Bibliography**.

Chapter 2

Cyber Crime

2.1 Introduction

Since the Internet's early beginnings in the 1960s [46], technological advances have been nothing less than phenomenal. As a consequence, society has also become increasingly aware of the risks of using Internet services [11]. However, society is largely uneducated about secure computing practices. Worse yet, the procedural and technological contributions for defending Internet services are unable to keep pace with the myriad of exploits facilitated through the Internet [78].

As society progressively becomes Internet-dependent, so too are criminal elements — whose aim is to exploit Internet services, applications and infrastructures. These criminal elements are widely referred to as cyber criminals, while their exploits are referred to as cyber crimes. Unsurprisingly, the incidence of cyber crimes continue to increase, given the growing uses and applications hosted on the Internet. To this end, classifications of several computer and Internet-based exploits have been identified — with variants of these known exploits continually identified and categorized accordingly.

The cyber crime classification utilized in this report is adopted from work published by Burden and Palmer [8], where cyber crime is categorized into *true* cyber crime and *e-enabled* cyber crime. In their work, true cyber crime is described as dishonest or malicious acts that would not exist outside of an online environment, or

at least not in the same kind of form or with the same impact. E-enabled crime is described as a criminal act known to the world before the advent of the World Wide Web but which is (now) increasingly perpetrated through the Internet.

The remainder of this chapter is structured as follows: in section 2.2 and section 2.3, we enumerate and expound on the different types of true and e-enabled cyber crimes. It should however be noted that we only present a discussion on classical cyber crimes within the said sections. For instance, malicious software dissemination and credit card misuse are discussed within sections 2.2.4 and 2.3.1 respectively. Nevertheless, we offer these sections as a backdrop for the content in latter chapters; the well-versed reader is therefore at liberty to only briefly examine this content. In section 2.4, we consider some of the characteristics and motivations of cyber criminals. Thereafter, a chapter summary is provided in section 2.5.

2.2 True cyber crime

In the following subsections, we elaborate on specific examples of true cyber crimes.

2.2.1 System intrusion (hacking)

Hacking can be described as the unauthorized entry of a malicious entity into private systems or confidential information. Hackers access network resources with the intention of perpetrating some form of fraud, or stealing confidential information for espionage. Worse yet, hackers could even render a system unavailable. In a paper titled *Gang culture in the online world*, Heron points out how hacking is increasingly no longer enacted by a single perpetrator. Heron [34] further indicates that hacking is emerging as the illicit and lucrative business of cyber gangs.

It is interesting to note some of the investigation methodologies employed by hackers in planning their attacks. Three significant precursors of an attack are, the acts of port scanning, social engineering and reconnaissance. Using a port scanning application, an attacker is able to determine (for a particular IP address):

- which standard ports or services are running and responding on the target

system,

- what operating system is installed on the target system, and
- what applications and versions of applications are installed on the target system.

The mentioned information is readily available from networked devices and can be obtained completely anonymously — which is ideal for any attacker.

Social engineering is arguably the mainstay of attack methods. With social engineering, a malicious entity typically masquerades as a legitimate entity — for instance, as an IT support technician. Using social skills and personal interactions, the malicious entity is then able to glean internal security-relevant information from the (trusted) internal source. This methodology exploits human nature and the well-known lack of end-user awareness.

At this point, using port scans, an attacker would have gathered the external architecture of the target system, while the internal network details would be the payload of a successful social engineering method. As an all-encompassing step, the attacker would perform some further reconnaissance — which is the general term used for information gathering. Therefore, even with the information at hand, the attacker would seek to further arrive at intimate details about the target system. This is achieved through acts such as eavesdropping or dumpster-diving — which involves (literally) scouring through rubbish bins or recycling boxes in search of private or sensitive internal information.

Equipped with these attack methodologies, hackers then study the target host's vulnerabilities and corresponding exploits. Using the gathered information, they then execute their exploits to achieve a malicious outcome.

Stoll [85] recounts how, over a ten month period (between 1986 and 1987), he and a team at the Lawrence Berkeley Laboratory detected and monitored a hacker, who initiated a systematic attack on 450 computers attached to the Defence Data Network (DDN). The DDN was operated by the Defence Advanced Research Projects Agency, and at the time, it interconnected approximately 30 000 computers. Furthermore, the DDN consisted of two major segments, Milnet and Arpanet. In his article, Stoll details the attacker's methodology, which focused on hosts on the Milnet segment.

The attacker systematically connected to Milnet hosts and attempted default account name and password combinations in order to access the targets. For example, for the VAX/VMS system, the attacker attempted the default account <<*system*>>, with password <<*manager*>>; the article further confirms that the attacker had a two percent success rate with these credentials. Fortunately, at the end of the ten month period, the attacker was caught and litigation was initiated.

Although the above account is somewhat dated, it is illustrative of how the enforcement (or the lack thereof) of minimal computer security standards has been inherited through the years; nowadays, with the use of password guessing applications, hackers are still able to exploit vulnerabilities resulting from default system settings.

2.2.2 Denial of Service (DoS) attacks

The aim of a denial-of-service (DoS) attack is to prevent legitimate system users from gaining access to, or using the required system services or functionalities [8]. Therefore, the outcome of a successful DoS attack is that systems become overloaded, or are simply unable to 'interpret' and hence process (received) input data. This results in servers going into a hang-state or even crashing. In the following paragraphs, we elaborate on some methods through which DoS attacks are achieved.

Amongst others, a DoS attack can be achieved using a buffer overflow — which is also known as a buffer overrun. A buffer overflow occurs when a program or process tries to store more data in a buffer than it was intended to hold. Given that program buffers are created to contain finite amounts of data, the additional information (which must still be stored) can overflow into adjacent buffers, thereby computing or overwriting the data stored within these buffers [30]. Buffer overflows may also occur as a result of programming errors. Irrespective of how the buffer overrun occurs, hackers are able to exploit this vulnerability by inserting malicious code into the buffers, thereby triggering malicious actions that invariable further the attacker's cause.

A DoS can also occur where servers are swamped with millions of invalid, or

bogus messages, to the extent that all available capacity on these servers is (over) utilized. These messages could also be invalid error messages; depending on the target network infrastructure, by system design, error messages must be logged by the recipient server(s). For example, a mail server may be able to accept and process a maximum of four hundred messages concurrently. However, a ten or twenty-fold increase in the number of messages to be processed is likely to strain server resources, thereby causing unexpected system behaviour. In other instances, DoS attacks are also achieved where the attacker sends over sized or malformed data packets to the target system.

A distributed DoS (DDoS) attack is a slight variant of the fore-mentioned DoS attack forms. A DDoS attack occurs where the attack is targeted at a single, or limited set of targets. However, these attacks typically originate from multiple sources — which invariably increases the 'potency' of the attack. For instance, an attack on a limited set of targets can occur where a high-volume website is distributed and possibly replicated to different data centers [47]. Lee et al [47] propose novel mechanisms for content distributed network (CDN)-hosted sites to withstand and deter DDoS attacks on CDN infrastructures. Furthermore, DDoS attacks are increasingly enacted using 'zombie' machines or botnets [43] — these are 'schools' of compromised network hosts that are used as surrogates for DDoS attacks. It is also not unusual for the owners of these computing devices to be unaware that their machines are compromised and are being used as staging points for cyber attacks.

2.2.3 Cyber vandalism

Cyber vandalism can be described as the graffiti of the cyber world. Such acts range from website defacements, to acts such as domain name hijacking.

Website defacement is achieved when a cyber vandal successfully compromises a web server and is able to substitute valid web content on the web server with invalid content. For instance, a hacker who penetrates a target (corporate) web server, could publish extremely crude content (such as profanities or even pornographic content).

Hackers have been known to achieve domain name hijacking through the exploitation of weak Internet Service Provider (ISP) administrative processes. This form of vandalism gained notoriety in April 2000, when attackers transferred over 50 companies' domain names to different web addresses [8]. This was a significant event because large corporations such as Adidas and Manchester United were the targets of this attack. In some instances, hackers have been known to even submit 'bad faith' complaints to domain name dispute bodies such as the Internet Corporation for Assigned Names and Numbers (ICANN). This is done in a bid to prevent domain name holders from using a domain name, even though they are known by that specific trademark and have legitimate rights to the name in question.

A somewhat more intricate form of cyber vandalism, called Domain Name System (DNS) poisoning, can also be achieved by hackers. In this instance, an attacker is able to alter DNS entries within a DNS server, thereby rerouting all valid web traffic to an invalid or malicious destination address. DNS is the directory service of the Internet, translating 'human-friendly' host names to network addresses. The impact of DNS poisoning is significant, due to the core function of DNS and the distributed and hierarchical nature of DNS databases. Therefore, a poisoned DNS record in a name server in China (for instance), could be promulgated to its adjacent DNS servers in Japan or Australia. This ripple effect is certainly undesirable and could potentially constitute a DoS attack.

Ultimately, cyber vandalism strikes at the image and branding of the affected organization, which may lead to significant financial losses due to minimized system usage. It could also negatively influence business aspects such as customer confidence — an intangible asset that is often difficult to regain after such incidents.

2.2.4 Malicious software dissemination

In this day and age, computer users are acutely aware of malicious software and the unexpected and undesirable manner in which such software influences computer operations. In particular, many computer users are aware of computer viruses. Although significant, computer viruses cannot describe all the malicious software in existence.

As a result, malicious software is classified into viruses, worms, or trojans — this classification is largely based on the software's behaviour.

Mishra and Saini [56] provide a formal definition for a computer virus; they describe a virus as *"a program that can "infect" other programs by modifying them to include a possibly evolved version of it. With this infection property, a virus can spread to the transitive closure of information flow, corrupting the integrity of information as it spreads"*. In fact, this definition is derived from much earlier works published by Fred Cohen [19, 20]. Cohen is widely credited as the *father of computer viruses*; on 3 November 1983, he conceived of the first virus as an experiment and presented the concept at a weekly computer security seminar. The name *'virus'* was thought of by Len Adleman [19].

In an article titled, *Computer Viruses: Theory and Experiments*, Cohen introduces the concept of a computer virus and examines their potential for causing widespread damage to computer systems. He also provides inputs on prevention and protection mechanisms against computer viruses. In a subsequent article on virus protection, Cohen [20] provides results of experimentation conducted on new virus protection mechanisms and explores the effectiveness of these mechanisms. It is interesting to note that, even in 1987, worms and trojans were discussed within these mentioned works of Cohen.

A worm is described as a program that spreads copies of itself through a network. A worm has the same malicious properties of a virus and can simply be seen as a type of computer virus [66]. The characteristic difference between a worm and a virus is that, the former operates through networks, while the later can spread through (any) digital mediums (but usually uses copied program or data files).

On the other hand, a trojan is described as a program that appears to be legitimate or generally interesting, yet contains additional functionality. When invoked, this additional functionality is intended to gain unauthorized system access or to cause some form of malicious damage. Pozzo and Gray [70] describe this type of malicious software as *insidious*, since trojans generally operate through legitimate access paths. Emm [25] provides a history of trojans, as well as an array of the types of trojans.

As a final elaboration on the impact of malicious software, a simplistic yet comprehensive illustration about the infection process for malicious software is provided in Figure 2.1.

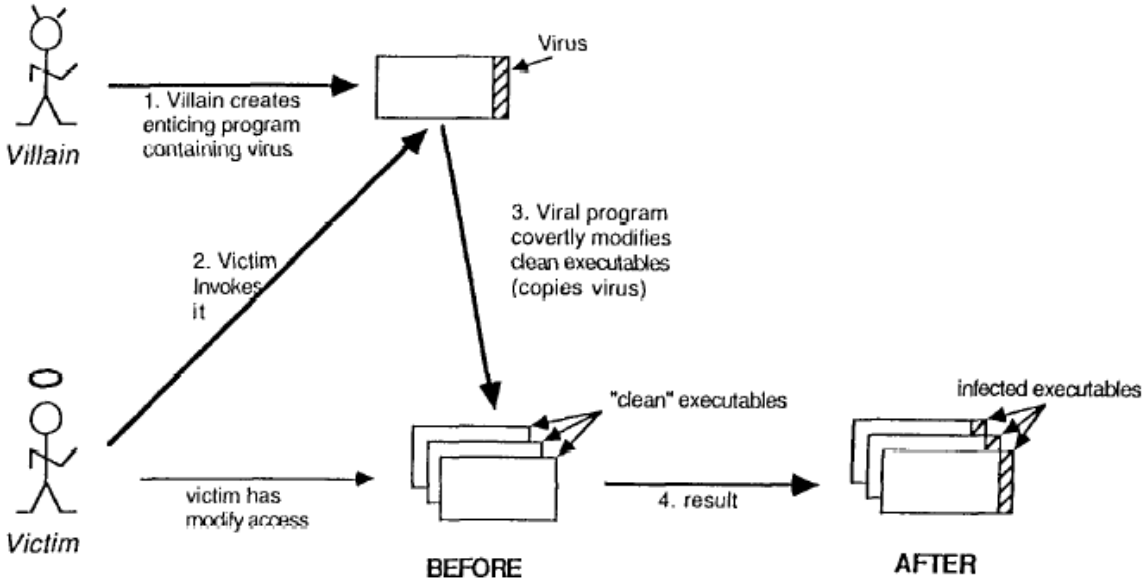

Figure 2.1: The process of a virus infection [70]

2.3 E-enabled cyber crime

In the following subsections we elaborate on specific examples of e-enabled cyber crimes.

2.3.1 Credit card misuse

In a world that is progressively migrating toward a cashless society, it is somewhat inevitable that credit card misuse and fraud would exist. Moreover, with the introduction of e-commerce applications and online banking facilities, credit cards are widely seen as a common medium for financial transactions within the cyber world. Although significant control mechanisms are enforced at a merchant level, as well as

key commerce applications and infrastructures, credit card misuse continues to go unabated.

From a non-technical perspective, the hands-on approach of dumpster diving is largely employed in the gathering of this crucial financial data. Effective data classification schemes would be highly beneficial in addressing reconnaissance methods such as dumpster-diving. Data classification mechanisms would stipulate how employees should dispose of corporate information, thereby minimizing the leakage of sensitive information. From a technical perspective, with the use of malicious technology such as websniffers and keyloggers, malicious entities are able to gather credit card information from unsuspecting computer users. In support of these attempts at online fraud, websniffers and keyloggers collect keystrokes and screen-shots in the theft of the required banking data [25, 30]. Thereafter, the credit card information, logged by these malicious programs, is then posted to specified locations and used at the attacker's discretion.

One of the major challenges with defending against this form of crime is the myriad of attack vectors through which network hosts can be infected with malicious technology (such as keyloggers). Such software can be downloaded and installed on a host machine due to web browser vulnerabilities, an end-user accessing malicious sites, or due to the invocation of malicious active code (such as Java code or ActiveX controls) on an infected website.

2.3.2 Information theft and misuse

Several forms of cyber crime can be included under the banner of information theft and misuse. For our purposes, only two of these forms are discussed, namely identity theft and phishing attacks. According to Chung et al [15], identity theft often appears in international trade and e-commerce, wherein a criminal may masquerade as a legitimate seller in order to obtain payment from buyers. In the case of phishing scams, attackers transmit large numbers of emails containing web links to webpages under their control. Once a victim clicks on the link, he or she is then directed to a seemingly legitimate webpage. It is here that the victim is instructed to enter their

private information — information that is subsequently emailed to a webmail address, or stored on the malicious web server for later collection by the attacker [58].

Phishing attacks are inherently difficult to address. This is largely due to the inadequacy of existing mail and web content filtering solutions; currently such solutions are unable to adequately detect or disallow such content from entering organizational networks. Furthermore, unaware end-users are continually enticed into supplying confidential information, under the guise that such information will be used for legitimate purposes.

2.3.3 Cyber obscenity or pornography

As a societal norm, pornography is largely shunned upon. This form of cyber crime refers to the promotion of pornographic material through the Internet [15]. More specifically, hard-core pornography, such as child exploitation images, is registered as a criminal offence under many legal texts. Therefore, all else equal, computer users found with such illicit material are prosecuted and sentenced under criminal statutes.

According to investigations conducted by the international High Technology Crime Investigation Association (HTCIA) [37], it has been determined that the majority of explicit child images in circulation (amongst the image consumers) are virtually the same. That is, there is a general retardation in the rate at which 'new' child exploitation imagery is added to existing image databases. As a result, the HTCIA has been able to generate hash sets (or unique 'fingerprints') for their database of child exploitation images.

2.3.4 Cyber piracy

In essence, cyber piracy constitutes a form of copyright infringement. This copyright infringement relates to software piracy, as well as video and audio piracy. Possibly the largest or most vocal activists against such crimes has been the major record companies. In recent years, cyber piracy has earned notoriety due to landmark litigations against peer-to-peer music sharing applications such as Napster. As an indication of the seriousness of such crimes, it is reported that in October 2001, the recording

industry attempted to sue Napster on the issue of liability, although Napster had ceased swapping copyrighted material in July 2001 [24].

Artists and media companies cite that there is significant intellectual property (IP) contained within the copyrighted material, and as such, this information should be protected. Understandably, the case against cyber piracy is that it stifles the economic drive for the production of new video or audio material. Additionally, vendors argue that this activity also negatively influences input costs and hence skews profit margins; variations is input costs are then invariably passed on to consumers. On the other hand, consumers believe that the majority of video and audio consumables are dramatically overpriced. Some consumers even argue that media companies and distributors gain super-normal profits, even with the prevalence of piracy.

It remains to be seen how artists, consumers and media companies will strike a balance between societal wants and business requirements.

2.4 Characteristics of cyber criminals

Publications such as [71], [8], [78], [57] and [44], are dedicated to reporting on the incidence and impacts of cyber crimes. In contrast however, publications on the motivations for attackers do not garner nearly the same interest from computer and information security professionals. Consequently, in the following subsections we consider some characteristics largely associated with cyber criminals.

2.4.1 Challenge

Pfleeger and Pfleeger [66] cite that, *"the single most significant motivation for a network attacker is the intellectual challenge"*. Furthermore, the authors state that *"some attackers enjoy the intellectual stimulation of defeating the supposedly undefeatable"* — in a telephonic interview, Kevin Mitnik (widely regarded as a notorious hacker) confirmed these assertions. In 1995, Kevin Mitnik was arrested by the Federal Bureau of Investigation (FBI). He was indicted for wire and computer fraud and served a five year prison term after pleading guilty to the mentioned charges. In a

telephonic interview with Cable News Network (CNN) reporter Manav Tanneeru [18], Kevin Mitnik provided some insight into his motivation for performing the mentioned acts; he stated that *"...in the past I was hacking for the curiosity, and the thrill, to get a bite of the forbidden fruit of knowledge"*.

In a different interview [23], when queried on his motivation for hacking, Jonathan James (the NASA hacker) responded that, *"It's intellectual. It stimulates my mind. It's a challenge"*. In 2000, Jonathan James was (also) arrested by US federal agents for stealing passwords, intercepting three thousand three hundred emails and stealing confidential information from thirteen NASA computer systems. He was fifteen years old at the time of the crimes and served a six month term in a Florida detention centre [33].

There are also hackers who, although actively seeking a challenge, do act with an ethical mandate. These hackers are referred to as penetration testers or white hats. Such people are usually employed by computer security companies or multinational organizations, in order to defend these organizations from cyber crimes. In instances where penetration testers act independently, they typically reveal system vulnerabilities only to the affected system vendors, thereby enabling the system developers some reprieve in order to remove or adequately address the identified vulnerabilities.

2.4.2 Fame

In some instances, hackers actively seek recognition for their exploits — possibly as a display of their technical or tactical prowess. The need for recognition also typically stems from the economic positioning or financial stature of ones target, which in turn signals the extent of ones triumph.

Given the nature of hacking and the consequences thereof, a degree of anonymity is required. Therefore, hackers (almost always) act under aliases, such as 'Kingpin', 'Zoz', 'mudge', 'DCFluX', or 'dildog'. Furthermore, although a hacker may perform numerous exploits, the hacker is unlikely to publicize their exploits through public mediums. Rather, their successes are typically publicized amongst their inner circles.

Hacker aliases are by no means glamorous. However, they do fulfil a chief-aim,

that is, to maintain hacker anonymity and yet achieve notoriety and acclaim amongst one's hacker peers.

2.4.3 Financial gain

With the prevalence of crimes such as, credit card fraud, advance fee fraud and identity theft, it is evident that financial gain is a significant (and common) motivating factor. At the launch of the National High Tech Crime Unit in April 2001, Bill Hughes, who was the Director of the UK National Crime Squad, made the following statement: *"Looking to the future the equation is simple — money is going electronic and where money goes so will organized crime"* [22]. Heron [34] suggests that some cyber criminals, specifically from Russia, engage in such activity due to unemployment. The author further explains that, due Russia's socio-economic climate, highly proficient IT specialists are lured to cyber crime by the relative high gains and low risks of these crimes — these assertion are also supported in work published by Govil and Govil [30] and McKenna [54]. It is also generally accepted that such exploits are largely performed from external sources (by outsider threats). However, the ever elusive insider threat is increasing in prevalence.

With specific reference to insider threats, Govil and Govil [30] indicate that the goal of cyber criminals is *"to sell or use valuable information for money"*. Any discussion on insider and outsider threats can also be extended to incorporate espionage. In broad terms, espionage occurs where an offender seeks confidential or sensitive information from a target organization. Historically, espionage is described from a political or military context, that is, one country spying on another country to obtain political or military advantage. Subsequently, the term industrial espionage or corporate espionage is used to explain such spying activity within commercial contexts, conducted for commercial rather than national security purposes [39]. Sunner [86] discusses the rise of malicious software, specifically where targeted Trojan code is utilized for industrial espionage. In their work, Power and Forte [69] examine the several cases of espionage within cyber contexts.

It is self-evident that financial gain underlies general criminal activity. Therefore,

one can only expect similar motives in crimes committed within digital environments.

2.4.4 Ideology

In the author's opinion, ideological motivations are complex for the following reasons: they are subjective and attacker actions tend to be extreme in nature, violent or even fatal. Prior to the World Trade Centre attack on September 11 2001, President Clinton had sought $2.8 billion to be spent on defence against chemical weapons, biological weapons and cyber-terrorist attacks [35, 84]; this is a clear indicator of the concerns surrounding this method of terrorism.

A distinction is drawn between *'hacktivism'* and *'cyberterrorism'* [66]. Hacktivism is described as operations that use hacking techniques against a target's network with the intent of disrupting normal operations but not causing serious damage. The authors further caution that cyberterrorism is more dangerous than hacktivism; cyberterrorism is described as politically motivated hacking operations intended to cause grave harm, such as loss of life or severe economic damage.

In wake of the September 11 2001 attacks and the heightened political tension globally, it remains to be seen how security mechanisms will, or can be adapted to effectively defend against ideology-based exploits.

2.5 Chapter summary

In this chapter, an overview of the new wave of criminal activity perpetrated by cyber criminals was provided, namely cyber crime. These cyber crimes were categorized into true and e-enabled cyber crimes, and the distinction between these crimes was explained. Thereafter, examples of cyber crimes, within the relevant categorizations, were detailed and discussed.

Undeniably, the acts perpetrated by cyber criminals always violate one of the three tenets of computer and information security, namely confidentiality, integrity and availability [90]. Confidentiality relates to the protection of information from unauthorized access. Integrity is the protection of information, applications, systems

and networks from intentional, unauthorized, or accidental changes. Availability is the assurance that information and or other system resources are accessible by authorised users whenever required.

Although not the chief-aim of this section, an extensive overview of attackers and their motivations was provided. Further classifications of attackers can be obtained in work published by Kjaerland [44]; in some classifications attackers are differentiated as pirates, browsers and crackers. Under another (more expansive) classification scheme, attackers are differentiated as pranksters, hacksters, malicious hackers, personal problem solvers, career criminals, extreme advocates, malcontents, addicts and irrational and incompetent people. Supplementary information regarding the psychology and motives of cyber criminals can also be sourced from [29].

It should however be borne in mind that, cyber crimes and criminal motivations are, by no means, limited to those mentioned in these preceding sections. However, it is widely understood that new types of cyber crimes are, more often than not, simply variations of the cyber crimes mentioned within this chapter.

Chapter 3

Computer Forensics

3.1 Introduction

Computer forensics exists to assist computer forensic specialists (CFSs) in the investigation of crimes commissioned through the use of computers or other digital devices — where evidence is digital in nature. Such investigations are conducted on storage media or network devices used in facilitating, or thought to be involved in the commissioning of cyber crimes.

For the most part, forensic investigations are undertaken in response to information security incidents and cyber crimes. Furthermore, computer forensic investigators are tasked to identify the sources or perpetrators of cyber crimes and to report on all digital evidence supporting or proving investigative hypotheses. During the investigative process, specialized software and hardware devices are utilized in analyzing the various levels at which digital data is stored.

As stated by Chawki [14], *"evidence is the foundation of any criminal case"*. Therefore, it is critical that the investigation life-cycle is conducted in a legally admissible manner. As such, computer forensic investigations follow a strict investigation methodology in order to maintain the integrity and credibility of all storage devices under investigation. That is, physical, technical and procedural measures for securing crime artifacts to evidential standards are implemented, thereby ensuring that investigative outcomes are able to withstand legal scrutiny.

The desired outcome of any investigation is the successful prosecution of the identified offenders. Therefore, similarly to other forensic sciences, computer forensics relates law and science.

The remainder of this chapter is structured as follows: in section 3.2, we provide a brief historical overview on forensic sciences. Thereafter, in section 3.3 we examine the field of computer forensics and identify the tasks typically performed by computer forensic specialists. By providing the fundamental concepts and terminology required within the field of computer forensics, section 3.4 highlights the level of detail to which forensic specialists are required to operate. A computer forensic investigation methodology and the constituents of a forensic process are provided and discussed in section 3.5. In section 3.6, we highlight some of the software tools utilized within forensic investigations. A chapter summary is then provided in section 3.7.

3.2 History of forensic sciences

There are several accounts of the history of forensic sciences. It is therefore challenging to identify a definitive source, or sources of this evolving class of science. Nevertheless, in this section we provide a summarized timeline of forensic science, focusing on the significant milestones within the development of this science.

In 44 BC, an ancient Roman physician named Marcus Antistius Labeo examined the fresh corpse of Roman emperor Julius Caesar (after his assassination). In spite of the twenty three stab wounds identified by Antistius, he knew which of these wounds proved fatal; he subsequently *announced* that only one wound to the chest caused the death of Caesar [21]. The manner in which Antistius relayed his findings is significant. This is because, seemingly, the term *forensics* was derived due to the fact that Antistius made his announcement before the *forum* — a term used in ancient Rome to denote a public place, where causes are judicially tried and orations delivered to the people. This is one of several accounts, indicating that forensic science has its origins in ancient Rome.

It is recorded that, in 1000 AD, Quintilian, who was an attorney in the Roman courts, provided the earliest known account of *investigative* forensic work — where

forensic science is used to prove or disprove legal arguments [38]. In a book titled *The Major Declamations* — which is an account of Quintilian's rhetoric — Quintilian recounts the case of a blind man who was accused of using his sword to fatally wound his father. In this case, Quintilian demonstrated that the blind man was actually framed by his stepmother. In spite of the sword being present in the wound at the time when the victim was discovered, Quintilian proved his case using the trail of bloody palm prints from the accused's bedroom to the victim's bedroom.

In 1686 the existence of contours (ridges, spirals and loops) in human fingerprints was discovered by Marcello Malpighi — an Italian professor of anatomy at the University of Bologna. Malpighi's work was extended in 1823 by John Evangelist Purkinji — who was also a professor of anatomy, at the University of Breslau. In his PhD thesis, Purkinji identified and discussed nine distinct fingerprint patterns and subsequently created a classification system based on these nine distinct points [21, 38].

In their works, neither Malpighi nor Purkinji made mention of the value of fingerprints as a tool for personal identification; this would only be discovered in 1880 by Henry Faulds, a Scottish doctor and missionary [6]. In an article published in the scientific journal, *Nautre*, Dr. Faulds discussed how fingerprinting could be used as a means for personal identification. He also suggested a method, which made use of printer ink, for obtaining fingerprints as residual evidence at crime scenes. Due to his application of fingerprinting within investigative contexts, Dr. Faulds is credited as the first person to recognize the value of latent prints left at crime scenes.

In more recent times, some of the revolutionary advances in forensic sciences have included Deoxyribonucleic acid (DNA) fingerprinting and ballistics. In 1984 Sir Alec Jeffreys developed a method for the identification of individuals from DNA — which is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms and some viruses. Sir Jeffreys made this discovery while a research fellow at the Lister Institute at the Leicester University; he dubbed his discovery as *DNA fingerprinting*. Given the widespread forensic application of DNA fingerprinting, also known as *DNA typing*, it is arguably viewed as the greatest single forensic discovery of the twentieth century.

In a somewhat different context, ballistics is utilized in the analysis of firearm

usage in crimes. In this science, the motion, behaviour, markings and gaseous effects on a bullet fired from a firearm are used as evidence during investigations. Volgas et al [92] indicate that the formal study of ballistics as a science only developed in the late nineteenth century — this development was largely attributed to the standardization of bullets and the development of high-speed photography. Ballistic sciences have subsequently expanded from their initial use, for the 'individualization' of bullets to a gun barrel, to the 'individualization' of bullets to a class of weapons, or cartridge and shell casings.

3.3 Computer forensic science

The term *Computer Forensics* was coined in 1991 in the first training session held by the International Association of Computer Specialists (IACIS) in Portland, Oregon [89]. Based on other accounts, it is apparent that computer forensic investigations existed in some form, even prior to 1991; as early as 1984, the Federal Bureau of Investigation (FBI) Laboratory, in conjunction with other law enforcement agencies, began developing programs to examine computer evidence [97, 14, 94]. To properly address the growing demands of investigators and prosecutors in a structured and programmatic manner, the FBI established the Computer Analysis and Response Team (CART) [94]. Based on Whitcomb's [94] account, although CART was unique to the FBI, its function and general organizations was subsequently duplicated within several law enforcement agencies within the USA and other countries.

In its formative years, some of the key challenges within the field of computer forensics have related to the definition of digital evidence, the lack of computer forensic expertise and the standardization of examination methodologies. However, with the advances in computing and network technologies, legal, forensic and law enforcement practitioners have been compelled to speedily address these challenges; to date these challenges are addressed to a significant degree. In the following paragraph we provide some of these developments.

Although several definitions of digital evidence exist, one of the widely applied definitions state that *"digital evidence is any information of probative value that is*

either stored or transmitted in a digital form" [94] — this definition, is adopted in the remainder of this report. Technical expertise within the field of digital forensics continues to increase fairly rapidly. Unfortunately, the forensic capacity within most institutions (law enforcement or otherwise) are unable to keep pace with the prevalence of cyber crimes, lending to the advancements in technologies. Lastly, computer forensic specialists (CFS) are expected to perform the following standard examination tasks [4]:

- **Protect** the subject computer system during the forensic examination from any possible alteration, damage, data corruption or virus infection.
- **Discover** all files on the subject system which includes existing normal files, deleted yet remaining files, hidden files, password-protected files and encrypted files.
- **Recover**, as much as possible, files that are discovered to be deleted.
- **Reveal**, to the extent possible, the contents of hidden files as well as temporary files used by application programs and operating systems.
- **Access** the contents of protected or hidden files if possible and legally appropriate.
- **Analyze** all relevant data found in special areas of the disk. The concept of special areas of a disk is explained later in subsection 3.4.3.
- **Document** the results of the analysis of the subject computer system. This analysis includes a listing of all relevant files and discovered file data. The documentation also provides an overview of the system layout, file structures and data authorship information. Further to this, any attempts to hide, delete, protect or encrypt information should also be revealed within the documentation.
- **Provide expert consultation** and/or testimony as required. This testimony would typically be required to substantiate or refute legal arguments in a court of law.

In an article published in 2004, Stephenson [83] states that *"conversations with seasoned practitioners suggest that digital forensic practice is in a period of redefinition"*. This is indeed true. With the convergence of technologies and the increasingly ubiquitous nature of digital information, forensic investigations are no longer only conducted on 'classical' media such as floppy disks, memory sticks, zip disks, hard drives, CDs, or DVDs. Rather, investigations are increasingly being conducted on devices such as routers, mobile phones, smart phones, digital cameras, personal digital assistants and random access memory chips [59, 93, 53].

The reader would agree that technological advancement is somewhat inevitable. For this reason, computer forensic science is likely to remain in "a period of redefinition". However, digital forensics is largely based on sound forensic foundations, which stands the field in good stead into the future. The only remaining challenge, could be within the adequacy of existing tool sets for forensic investigations.

3.4 Fundamentals in computer forensics

In many instances, the success of a computer forensic investigation may hinge on more than the investigative experiences and instrumentation expertise of a CFS. In fact, there may be a deep reliance on the forensic specialist's knowledge of the host operating system (OS) and the associated file system features of the subject computing system. With such knowledge, a CFS gains insight to the mechanisms by which a host OS manages the storage and removal of data within the subject storage media. Such information is invaluable for leading investigators to special areas on storage media; these special areas often contain relevant forensic evidence.

In the following subsections we introduce some of the underpinning concepts and terminologies utilized within the field of digital forensics. In so doing, we provide the reader with some perspective of the level of detail to which CFSs often operate. Given the prevalence of Windows-based operating systems (OSs), the terminology concentrates on these OSs. However, the rationale is extendible to other OSs, such as Unix and its more recent derivatives.

3.4.1 Disk organization

It is widely understood that bits are stored on hard disks in blocks of data called *sectors*. In order for the operating system to manage storage space on drives, information is written to one or more contiguous groups of sectors called *clusters*. Under Windows-based operating systems, clusters consist of fixed length blocks of bytes. Therefore, when the OS stores information, it is written to particular clusters and not sectors per se — this is due to the efficiencies associated with tracking clusters, as apposed to sectors.

A key OS system feature, relating to the tracking of these clusters is achieved using a *file map* or a *file allocation table (FAT)*. In a file map, each disk block is represented by a single map entry. Similarly, the FAT is a file containing an association between sectors and their physical location on the hard disk [31].

The number of sectors within a cluster is also dependent on the volume of the hard disk in question. For instance, given the same disk capacity, cluster sizes under the FAT12, FAT16, FAT32 or NTFS filing systems may vary considerably. In Table 3.1 we provide a generic association between disk sizes and cluster sizes.

Table 3.1: Disk size and cluster size association [88]

Storage capacity	**Number of sectors**
High density 3.5 inch floppy diskette	1
High density 5.25 inch floppy diskette	2
16MB - 127MB logical hard drive partition	4
128MB - 255MB logical hard drive partition	8
256MB - 512MB logical hard drive partition	16
512MB - 1024MB logical hard drive partition	32
1024MB - 2048MB logical hard drive partition	64
2048MB - 4095MB logical hard drive partition	128

In the following section we briefly discuss filing systems and their significance within a forensics context.

3.4.2 Filing systems

Amongst other things, the filing system in an OS is responsible for file directory management and file storage on electronic media. Therefore, when an end-user 'commits' information to secondary memory or disk, the filing system is responsible for assigning this data to clusters on the storage area. The challenge, however, lies in the fact that file sizes are variable. As a result, filing systems employ dynamic disk allocation techniques in order to store and manage used disk space and free disk space. In the following subsections we describe some of the techniques used by OSs in allocating file blocks to disk and retrieving these.

As a precursor, we explain the terms *user file directory (UFD)* and *master file directory (MFD)* — which are used in the following subsections. A UFD contains the names and locations of a single user's files. For each user in the system, a MFD contains a pointer to a UFD for that user.

3.4.2.1 Contiguous files

In the instance where the blocks of a file are stored in contiguous clusters on disk, the UFD entry for the file points to the first block of the file and also contains the length of the file. This is the simplest method of file management. However, it suffers from fragmentation; as files are created and deleted, free disk space becomes broken up into small 'pieces', none of which may be large enough to hold any information by themselves. This phenomenon is also known as *internal fragmentation* [82]. It is at this stage that *compaction* or *defragmentation* is required. During the act of compaction, files are moved around in order to consolidate free space into one or more usable areas. Under certain conditions, there is a known problem relating to the efficient allocation of space. This space allocation problem is typically manifested when the final size of a file is unknown at the time when the first block of the file is allocated [50].

An illustration of this filing technique is provided in Figure 3.1.

Due to the shortfalls in disk management techniques such as this, forensic specialists often leverage this to discover evidence within fragmented disk areas — we

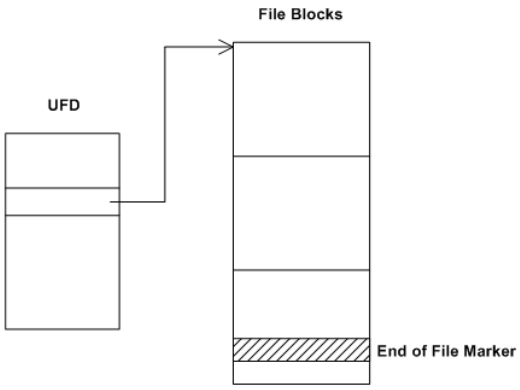

Figure 3.1: Contiguous file blocks [50]

elaborate on this in section 3.4.3.

3.4.2.2 Block linkage

With the block linkage technique, a few bytes within each block of a file are used as a pointer to the next block within the file. The last block within the file contains a *null pointer*, which is a marker indicating the end of a file block. In this instance, the UFD entry for the file points to the first block in the chain representing the file. This linking mechanism is depicted in Figure 3.2.

Access to file blocks is achieved in a sequential manner. The challenge in this mechanism relates to the number of disk accesses required in determining the end of a file. This is especially disadvantageous when a file is to be deleted and the currently occupied space is to be reassigned to a free list. In addition, the alterations to pointers in order to achieve the end-goal are reliant on the knowledge of the end of file. To compensate for this shortfall, the UFD entry is often extended to point to the last block in the file [50].

Furthermore, with this block linkage method, damage to one block (and the pointer contained within it) could lead to damage of the entire file system. It is for reasons such as this, that forensic specialists are required to record the present state of a target machine, including existing volumes and partitions and Basic Input Output System (BIOS) settings. In the instance where a filing system is corrupt, it

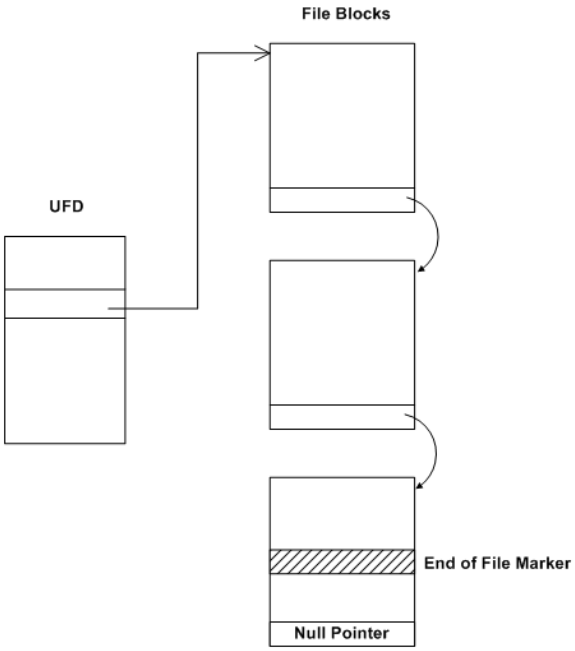

Figure 3.2: Linked file blocks [50]

is mandatory that this is reported during the investigation cycle.

3.4.2.3 File map

In the file map method of file linkage, the state of the disk is recorded in a file map or file allocation table, in which each disk block is represented by a single map entry. As a slight variation, the UFD entry for a file points to the location in the file map representing the first block in the file. This location then points to the location in the file map representing the next block in the file. In this manner, the entire file is mapped accordingly. The last block in the file is represented by a null pointer. In Figure 3.3, the file occupies blocks 3, 6, 4 and 8 on the disk.

To some extent, the effectiveness of this disk management technique is dependent on the size of a map entry. In small disks, a map entry can be as little as 12 bits, while this could grow to 32 bits on larger systems. This is significant because map entries may also contain additional (redundant) information, such as a unique file identification number — which is typically implemented to enable data recovery after

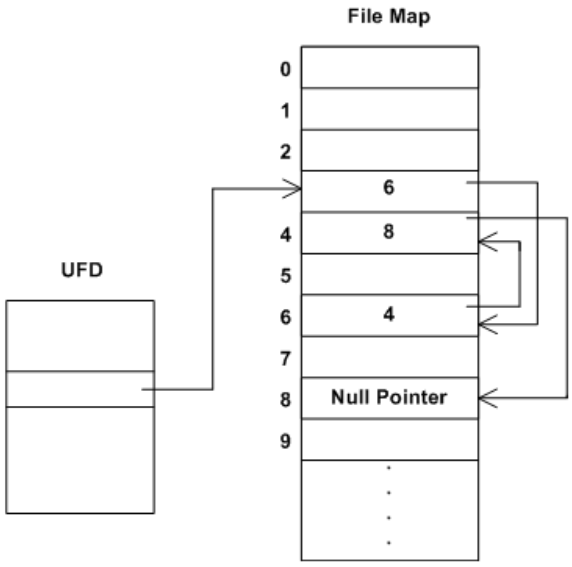

Figure 3.3: File map [50]

a system failure [50].

File maps are typically large; this is attributed to the considerable data contained within them. For this reason, they are stored on disk and their contents are brought into main memory (a block at a time) when required. In practice, in order to defend against the destruction of the file map, two or more copies of the map are kept at different areas within the disk [50]. With the use of sophisticated instrumentation, a forensic specialist is able to recover vital digital evidence, such as a file map; in some instances, using file map information, it may even be possible to interrogate and recover specific digital evidence from damaged hardware.

3.4.3 Ambient data

To the common system end-user, it is perceived that data becomes non-existent and no longer accessible when deleted. This is however not the case; data may remain in some form, or on certain areas of storage medium even after deletion. The term *ambient data* is used to describe the special areas on storage media, where data becomes stored even after deletion. From a computing perspective, the existence of data is dependent on the behaviour of the filing system in question; with this knowledge,

forensic specialists therefore interrogate these special areas of media during their investigations.

In the following subsections we elaborate on some special areas of disk, where ambient data is typically found.

3.4.3.1 Unallocated space

When data is deleted, it is the *reference* to such data within the file map (or file allocation table) that is actually deleted [82]. That is, the data may still exist on the storage media, however, the OS has no means by which to retrieve this information. Furthermore, when a file is 'deleted', the space occupied by the file becomes 'eligible' for allocated to new data — the space is typically labelled as 'free', in order for the filing system to assign new data to this area. Therefore, until such time that the storage area is occupied with new data, or overwritten, the 'deleted' data remains present within the respective clusters on the storage media.

3.4.3.2 File slack

When files are created, their lengths may vary depending on their contents. Furthermore, depending on the filing system within the OS, a file may not necessarily fully occupy the media space allocated to it [31, 82, 50]. In such instances, the space from the end of the file, to the end of the last cluster allocated to the file remains available. This space is commonly referred to as *file slack*. Such areas are investigated because they may contain previously created information.

3.4.3.3 Operating system and application-created files

One of the techniques by which an OS effectively manages system and application-created processes is by creating intermediary files. These files are referred to as *swap* files and the technique is commonly referred to as *swapping*; unsurprisingly, swap files often contain essential digital evidence, hence the reason why it invites interest from forensic investigators. Examples of operating system files and application-created files

include boot files, swap files, cache files, registry files, temporary files and history or log files.

Swapping enables a computer to execute programs and manipulate data files that are larger than main memory. Depending on the data required by the operation at hand, the OS copies the maximum allowable amount of data into main memory, leaving the 'unnecessary' data on the disk. When data from disk is required, the OS then exchanges portions of the data within main memory with the data recently retrieved from the disk [82].

3.5 Computer forensic investigation methodology

Computer forensics explicitly deals with the [99, 41]:

- Acquisition
- Preservation
- Identification
- Extraction
- Analysis and
- Documentation of computer evidence

Acquisition relates to the seizing of digital evidence at the scene of a digital crime. A requirement after evidence collection is the **preservation** of such evidence. From a legal stand-point, it is mandatory that a chain-of-evidence is maintained to ensure the protection of evidence from any forms of tampering. A chain-of-evidence refers to documenting the identity, custody and control of evidence from the point of collection, through to (and beyond) the point of presentation at a court of law [98, 53].

As an inherent step within the analysis phase of an investigation, a computer forensic specialist is expected to identify all possible forms of digital evidence on storage media attached to the computing devices under investigation. For instance,

this **identification** step could include the discovery of system files, hidden files, or password-protected files on the storage media.

Best practice requires that investigations be conducted on replicas (images) of the storage devices under scrutiny. Replicas are created through a process called *imaging*. The **extraction** process further ensures the integrity of all source evidence. Once the images are created, the **analysis** then begins. Thereafter, the results of any investigation are **documented** and reported on, as required.

In brief, we have discussed these widely accepted tenets of computer forensics. This is done to demonstrate the integration between these tenets and the tasks (provided in section 3.3) typically performed by forensic specialists. A formal forensic process is depicted in Figure 3.4, illustrating the cyclical nature of the investigative process. The data acquisition and authentication phases are compulsory predecessors of the analysis phase. On the other hand, the evidence documentation phase is performed in tandem with each phase of the process. This ensures that, for reference purposes, the investigative activities and evidence *fingerprints* are recorded and maintained.

In the following subsections we elaborate on the individual stages within the forensic process depicted in Figure 3.4.

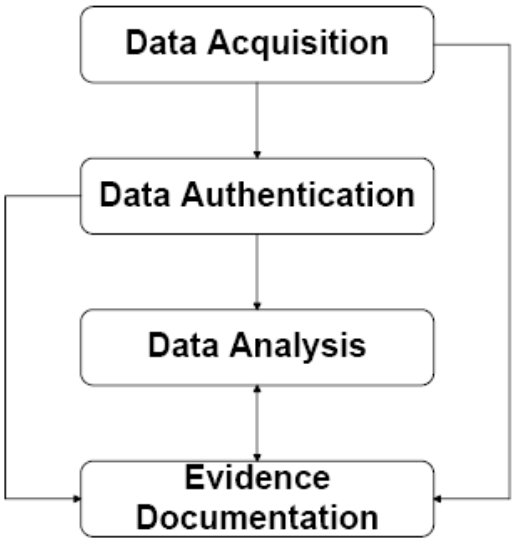

Figure 3.4: Stages within the forensic process

In principle, an investigation must be repeatable and verifiable. That is, an independent third party must be able to examine the investigative processes, execute them, and arrive at the same results. Ultimately, strict forensic methodologies uphold this principle, thereby ensuring the legal admissibility of evidence presented by an investigator.

3.5.1 Data acquisition

Forensic specialists do not (and should not) conduct their investigations on the source storage media under investigation. For this reason, investigations are conducted on images, acquired through the process of imaging. Imaging is defined as, the physical sector-by-sector and cluster-by-cluster copy of a storage medium and the compression of the image into a file for forensic purposes. Therefore, disk imaging utilities perform a bit-wise copy of digital data of a source medium to a destination medium; this copying could be performed on a file, a part of a file, swap files, logical drives, physical memory, or an entire physical disk.

Ideally, the destination medium is identical or nearly identical to the source medium. It is however reported by Lyle [51] that, an image file does not necessarily require the same disk geometry as the source storage device(s); it is possible to simulate the source disk geometry, if it becomes necessary to boot into the acquired image. Furthermore, forensically valid differences do occur when the source and destination media are not the same size, or if partitions from the source media must be relocated on the destination media in order that the destination partitions are accessible [51].

In some instances, disk imaging may occur against faulty source media. In such instances, many imaging tools are able to record disk I/O errors during the imaging process. It should be noted that disk errors are also recorded within the investigation log.

3.5.2 Data authentication

Data authentication is required once an image of the source evidence is acquired. The objective of the authentication phase is to verify that the acquired image file is indeed a replica of the source media. This is achieved using verification mechanisms known as *hash functions*.

A hash function, H, is a transformation that takes an input, m, and returns a fixed-size string, which is called the hash value, h [46]. Therefore, h is the result of the hash function applied onto input m. Hash functions are the foundation upon which authentication tools verify the integrity of original media and the resulting image files.

In the following sub-sections, we provide an overview of the Message Digest 5 (MD5) and the Secure Hash Algorithm (SHA), which are the most commonly used authentication mechanisms within forensic tools.

3.5.2.1 MD5

The MD5 algorithm was developed by Professor Ronald L. Rivest of the Massachusetts Institute of Technology (MIT) [46]. The algorithm guarantees the integrity of an input message through the creation of a 128 bit message digest (hash value). When the MD5 algorithm is applied onto a file, the resulting message digest is said to be as unique to that file as a fingerprint is to a human.

It is conjectured that it is "computational infeasible" for any two data inputs to have the same message digest [46, 72]. According to Rivest, *"the difficulty of coming up with two messages having the same message digest is in the order of* 2^{64} *operations, and that the difficulty of coming up with any message having a given message digest is in the order of* 2^{128} *operations"*. Guarantees such as these make MD5 a credible hashing algorithm.

Detailed specifications relating to the MD5 algorithm can be obtained at [72].

3.5.2.2 SHA

The design principles of the Secure Hash Algorithm are based on those of MD4, the predecessor of MD5. Therefore, the 'guarantees' associated with MD5 are also associated with this algorithm. SHA produces a 160 bit message digest when an input message of 2^{64} bits is provided. In recent years, SHA1, which is an improvement of SHA, is commonly available within forensic tools. Detailed specifications relating to the SHA1 algorithm is published at [40].

3.5.3 Data analysis

During the data analysis phase, forensic specialists interrogate image files for all forms of digital evidence, with the aim of supporting or refuting the investigative argument at hand; it may also occur that, information revealed during analyses contribute towards the development of investigative hypotheses [28]. Some of the information that is typically sought after include telephone numbers, network addresses, email addresses, names of individuals and various file names.

Data analysis is arguably the core of any investigation. Under the banner of an investigation methodology, investigative tasks are performed cyclically in arriving at definitive conclusions.

3.5.4 Evidence documentation

The objective within the evidence documentation phase is the documentation and presentation of all investigative analyses conducted; the chain-of-evidence is also typically presented during this phase. A desirable outcome within this phase includes the 'translation' of technical terminologies within the reports. Therefore, due to the varied target audiences of such reports, investigators are often required to construct their reports in a manner that is understandable by investigators and legal entities alike. For instance, this formatting could include the conversion of hexadecimal or binary information into readable characters, or the conversion of image file formats into formats that can be used as input to other analysis tools.

The presentation of information could very well 'make or break' a case. Therefore it is critical that the necessary attention is afforded to this phase.

3.6 Computer forensic tools

In the early beginnings of computer forensics, 'First Generation' forensic tools were typically command line based and often served a single purpose. As a result, two distinct branches of forensics have risen, network forensics and host-based forensics. Network forensics concerns itself with the investigation of data packets across networks and the hardware devices used to facilitate data transfer. On the other hand, host-based forensics deals with the investigation of physical surfaces of storage media [22].

The reader will note that, up to this point, much of the content within this chapter focuses on host-based forensics. However, the procedural and technical requirements (and considerations) within host-based forensics are indeed relevant and applicable within network forensics.

It is within the 'Second Generation' of tools that the overarching characteristics of forensic tools is revealed. These forensic tools are differentiated by cost, complexity, functionality and the OSs they support [59]. These tools are typically multi-functional and GUI-based. In many instances, the multifaceted nature of a tool is what lends the tool to complexity. For example, a tool's complexity may be a result of algorithmic complexity. However, a tool's complexity may also be exhibited in the tool's ease-of-use. Cost is the final distinguishing factor; some of the market leading tool-sets cost several thousand dollars, while other (equally effective) tools are freely available for download from the Internet.

There is presently a call for a 'Third Generation' of forensic tools; the expectation from such tools include automated image analysis, streaming media analysis, a multi-user environment and on-the-spot or "live" forensics [41]. Evidently, this generation of tools would enable the cyber crime scene to be visited remotely and allow mission critical systems to remain operational [22].

Ultimately, forensic tools and forensic tool kits (sets of forensic tools) are utilized

within the relevant phases of the investigative process.

Examples of forensic tools and their characteristics are provided in Table 3.2. However, complete information regarding the mentioned tools is obtainable at their associated reference.

Table 3.2: Computer forensic tools

Forensic Tool	**Operating System**	**Functionality**	**User Interface**	**Time**
dd	Windows, Unix	Imaging	command line	1st Gen
EnCase (Forensic Edition)	Windows, Linux, Solaris	multi-functional	graphical	2nd Gen
SafeBack	DOS, Unix, Windows	Imaging	DOS-based	1st Gen
ByteBack	Windows, Unix	Imaging	DOS-based	1st Gen
grep	Windows, Unix	Analysis	command line	1st Gen
EnCase(Enterprise Edition)	Linux,Windows, Solaris	multi-functional	graphical	1rd Gen
DriveSpy	DOS, Unix, Windows	Analysis	DOS-based	1st Gen
The Sleuth Kit	DOS, Unix, BSD, MAC, Windows	multi-functional	command line	2nd Gen
The Coroner's Toolkit	Unix	multi-functional	command line	2nd Gen
Forensic Tool Kit	DOS, Unix, Windows	multi-functional	graphical	2nd Gen
Ultimate Tool Kit	DOS, Unix, Windows	multi-functional	graphical	3rd Gen
TcpDump	BSD, SunOS, Linux,Windows	network analysis	command line	1st Gen
Snort	Windows, Unix	network analysis	command line	1st Gen

3.7 Chapter summary

This chapter provided an overview of the field of computer forensics. As a starting point, a historical overview of forensic sciences was presented — this timeline established the development of forensic science, through the significant contributions of academics, medical practitioners and practitioners within law enforcement.

With the convergence of technologies and the growth of embedded computing, computers are no longer (simply) discrete and identifiable pieces of machinery. Nowadays, a 'computer' can be any device containing a computing system. In spite of these developments, the fundamentals of computer forensics remain unchanged. As such, we presented some fundamentals within this field — it is these fundamentals that inform investigators on how, what and where to interrogate digital sources for digital evidence. It can be noted that a synopsis of digital evidence was withheld in this chapter; this was intentionally omitted, largely due to the broad number of digital artifacts that are considered as digital evidence. Nevertheless, when necessary, examples of digital evidence was periodically provided throughout the chapter.

Given the investigative underpinnings of forensic sciences, an investigative process, specific to computer forensics was provided. Furthermore, the investigative tasks and considerations within this process were discussed. The characteristics of forensic tools were then considered and examples of these tools were provided.

Chapter 4

Logging and Log Correlation

4.1 Introduction

The common thread within all forms of forensic science is embodied within Locard's Exchange Principle — this is a forensic principle expressed by Dr. Edmond Locard in 1923. Locard's principle states that, *"with contact between two items, there will be an exchange"*. In subsequent elaborations, Locard mentions that, *"it is impossible for the criminal to act, especially considering the intensity of a crime, without leaving traces, where these traces bare mute witness against the perpetrator"* [22]. Although this principle was derived in relation to investigations involving physical evidence, where perpetrators come into physical contact with the crime scene, it remains relevant and reliable within the digital age.

Within digital contexts, log files are a significant form of trace evidence. In particular, Rogers [73] indicates that *"the eyewitness of today and tomorrow may be a computer generated 'log file' "*; it is comments such as these that amplify the significance of log files within digital investigations. In the author's opinion, few other evidentiary artifacts hold as much significance to digital investigations as log files. In fact, system administrators and computer forensic specialists routinely make use of log files as an essential part of their incident response services.

The proposed model's reliance on log evidence is evident within the subsequent chapters. Therefore, the rationale for this chapter is to examine the usefulness of

logging and log correlation towards forensic investigations and the overall threat identification process. Specifically, this work establishes the role of log evidence within the forensic evidence management system (FEMS).

The remainder of this chapter is structured as follows: we elaborate on the fundamentals of logging in section 4.2. In section 4.3 we discuss some prevailing correlation techniques, namely the rule-based, fuzzy-based and model-based correlation techniques. Thereafter, the primary hindrances to effective log correlation are presented in section 4.4. In section 4.5, an overview of various works within the field of logging and log correlation is presented; the characteristics of log files for effective correlation are also provided in this section. We then conclude the chapter with a chapter summary in section 4.6.

4.2 Fundamentals of logging and log correlation

If properly programmed and configured, all information technology (IT) and network objects are capable of producing logs to reflect activity on these infrastructures [79, 26]. Event auditing capabilities are typically enabled with respect to the sensitivity or classification of data, systems, or applications to be protected; in the event of an incident, log files assist with the reconstruction and sequencing of the activities leading to the incident [60]. To this end, one understands the necessity for custom applications, web servers, FTP servers, mail servers and access control mechanisms to generate logs.

There are two basic types of logs, system and network logs [26]. System logs typically reflect application behaviour on the host, or behaviour with the underlying operating system. On the other hand, network logs are generated by devices or applications responsible for the inspection of network traffic [60]. System and network administrators would agree that logs are difficult to maintain and monitor [79, 45, 3], especially considering that audit data is typically large and verbose. Furthermore, given the multi-layered defence mechanisms for networks, these trace logs are increasingly becoming crucial witnesses to events.

A fundamental challenge is that audit data is useless unless it is regularly reviewed [77]; as alluded to earlier, this review is by no means a simple task. Moreover, the disparity between audit data formats (amongst differing technologies) restricts the effectiveness of many automated log correlation efforts [16] — log correlation techniques and the hindrances to log correlation are explored further in sections 4.3 and 4.4 respectively. At this stage we provide some clarification on the concepts of log analysis and log correlation as follows: log analysis involves the inspection, or review of individual logs. On the other hand, log correlation is defined as the improvement of the assessment and threat identification process, by not only looking at individual events, but at their sets, bound by common parameters [26]. Therefore, the combination of log analysis and log correlation can be deemed invaluable within digital forensic investigations.

At the heart of every correlation system is a correlation technique, implemented within a correlation engine. In the following section, the components, strengths and weaknesses of some frequently used correlation techniques are provided.

4.3 Log correlation techniques

In this section, the rule-based, fuzzy-based and model-based correlation techniques are presented. A highly structured log file, such as a web proxy log, is utilized to illustrate each correlation technique.

4.3.1 Rule-based correlation

Rule-based correlation techniques are the most commonly used method for event correlation. This is largely because their implementation is simple and rule generation is intuitive. In this approach, the task of event correlation is achieved through the knowledge and expertise within a rule-based correlation (RBC) system [61]. Therefore principles within machine learning and pattern recognition are embodied within this correlation method.

A typical RBC system consists of three fundamental components:

- A working memory,
- A rule base, and
- A correlation algorithm (or engine).

The working memory consists of facts about the environment. The rule base represents knowledge about facts that can be inferred, or can specify actions to be taken, given any particular fact within the working memory. This rule base typically consists of *if-then* statements and other predicates [10]. For instance, facts within the working memory could be:

1. *web proxy 223.1.3.27 return code 200 = O.K*,
2. *web proxy 223.1.3.27 return code 400 = Bad request.*

Fact 1 is interpreted as: "*the client request was successfully served by the proxy server*". While fact 2 is interpreted as: "*the client request could not be understood by the proxy server*". Rules within the rule base could be:

1. *if ((223.1.3.27 return code=200) or (223.1.3.27 return code=400)) then add_to_memory (src_ip_addr, dst_ip_addr, hostname)*
2. *if (223.1.3.27 return code=200) then add_to_memory (Print hostname of successful access)*

These rules are interpreted in the same manner as facts 1 and 2 above.

The correlation algorithm is the mechanism that actually determines inferences. Inferences refer to the actions that occur as the correlation engine makes passes over the working memory and the rule base. Therefore, after the correlation engine makes a first pass over the working memory and the rule base, the working memory becomes populated with new facts — these new facts may result from actions taken in the rule base, or other agents within the environment adding new facts to the working memory. On subsequent passes of the correlation algorithm, other rules may be initiated as a result of the new facts within the working memory.

However, there is a fundamental drawback to this correlation approach; the rule-base must be accurately generated so that inappropriate inferences are less likely to be issued by the correlation algorithm. As a result, the construction of correlation rules almost always requires considerable effort. Burns et al [10] describe how data mining techniques are applied onto logs containing alarm data in an effort to accurately generate correlation rules. During this effort, infrequent but highly likely associations are discovered and added to the rule-base.

Ultimately, this correlation mechanism is recommended for use within small, well understood and non-changing environments.

4.3.2 Fuzzy-based correlation

The fuzzy-based correlation technique is a variation of the rule-based technique.

We know that computers make use of numbers during computations. However, fuzzy logic enables the use of vague values that mean nothing to computers but mean something to humans. In all instances, this approach to decision making employs types of probability within mathematics and memberships of different sets [32]. This type of logic makes use of a fuzzy-inference rule base to determine which fuzzy membership functions best satisfy the conditions for correlation.

Fuzzy-inference rules are derived from words or sentences in a natural or artificial language. This makes additions or modifications to the existing rule base simpler to achieve. For instance, Aboelela and Douligeris [1] use fuzzy logic to determine the chronological relationship between events; the authors achieve this through comparisons of the origination and termination times of an alarm. An example of a fuzzy-inference rule would be: *if (request(T_1) < request(T_2)) then serve the requester at T_1 prior to serving the requester at T_2*. T_1 and T_2 represent the time when independent requests are received by the web proxy (223.1.3.27). In this instance the imprecision, or fuzziness, inherent in the timing of events is taken into account.

The concept of fuzziness is used within correlation engines to emulate intuition, forecasting and intelligent guessing. Ultimately, it enables computers to incorporate imprecision into correlation engines [32].

4.3.3 Model-based correlation

With a model-based correlation approach, software is used to represent network objects within an environment. This software is referred to as a model. A model could either represent a physical entity or a logical entity. Physical entities include hubs, routers, or computer systems. Logical entities include LANs, domains, or services. Whenever a model represents a physical entity, direct communication between the model and the entity takes place via some predetermined protocol.

All models contain descriptions. These descriptions include model information such as attributes, their relations to other models and their behaviour. These descriptions can be likened to descriptions of objects within the object-oriented paradigm. Examples of attributes for device models include network and hardware addresses. Common relations among device models include, *'connected to'*, *'depends upon'*, *'is a kind of'* and *'is a part of'* relationships. Model behaviour is determined through *if-then* statements, through the incorporation of model attributes and their relationships. Ultimately, with this technique, event correlation is a result of collaboration, or the collective behaviour of relevant models within the environment. Similarly to rule-based correlation, this approach is simple to implement and is therefore predominantly applied within small domains; within large domains, the numerous interactions between the models would hamper the performance of event correlation.

In this section an overview of existing correlation techniques was presented. In section 5.3 we indicate the correlation technique that is selected for inclusion within the proposed model. In the following section, characteristic differences occurring within log files are presented. These differences typically hinder correlation efforts.

4.4 Hindrances to log correlation

We begin this section by presenting the core differences that occur within log files, namely time-based and content-based differences.

For all intents and purposes of this section, let us assume the integrity of log files from the point of acquisition, through to the point of storage. Therefore, analyses

and correlation are assumed to be conducted on reliable and accurate log data — where reliability refers to the consistency of the measuring or recording process, while accuracy refers to the difference between the true value and the measured or recorded value [13].

4.4.1 Time-based differences

The Open System Interconnection (OSI) model describes the interactions between the seven layers within any network [66, 46, 31]. Furthermore, due to the interactions between these layers, data errors and data losses are sometimes introduced [13]. Therefore time-based differences are typically expected within networked environments. In general, time-based differences within log files occur as a result of the following:

- Differing system-clock settings,
- Time zone bias, and
- Differing system-clock speeds.

Differing system-clock settings lead to skewing, which reveals a lack of system-clock synchronization amongst log-enabled machines within a network. Differing time zones (or time zone bias) also poses a significant challenge for event correlation. Boyd and Forster [7] describe an example where, within an NTFS [1] partition, a simple text file was created. Thereafter, the time zone bias of the machine on which the text file was created was changed. The effect was that the text file's created and last modified times were immediately altered; understandably, such changes negatively influence the outcome of forensic examinations. However, Boyd and Forster highlight that registry adjustments to the time zone bias can rectify this problem. Finally, differing system-clock speeds lead to stretching or shrinking.

Reasonable guarantees must be associated with the dates and times that events are registered within a log file. Hence time-based differences manifest in the absence

[1] New Technology File System is the standard file system for Windows NT and its descendants.

of such guarantees. Time-based differences can be prevented, or at least curbed with the introduction of a trusted time source within the networked environment (using the NTP protocol). Therefore a trusted time source is a key requisite for the establishment of a sound forensics capability.

4.4.2 Content-based differences

Content-based differences can be classified into the following categories:

- Expected differences, and
- Unexpected differences.

As an example, expected differences between logs would occur when an application uses machine A but not machine B. For instance, if machines A and B are web proxy servers, then all the entries in the web proxy log on machine A, that have been served from machine B, should occur in both machines log files (if replication is taken into account). However, B will also contain material served to other machines and A will contain web requests served from cache, or sent to other machines. Unexpected differences between log files would typically occur as a result of data corruption, network failure, loss of data, application or system failure, or an unidentified error, during or at the time of logging. The intentional or unintentional deletion or introduction of log entries may also result in unexpected differences between log files [13].

In the following section we review some specific contributions within the field of logging and log correlation. In so doing, the facets, hindrances and (some) advances in this field are explored further.

4.5 Related work

Related work within logging is provided in the subsequent paragraphs as follows: Information that is typically logged within IT environments are provided. Fortes [26] three fundamental characteristics of log files for effective correlation are then presented — of these three characteristics, log file integrity is arguably the most integral

for correlation and evidential integrity. Leading on from Fortes work, information on mechanisms for maintaining the integrity and confidentiality of log files is also presented in this section; these mechanisms are further highlighted through the work of Kroon and Olivier [45]. Finally, Kenneally's [42] legal considerations for the appropriate use of log files is explored.

4.5.1 Purpose and content of log files

Historically, event correlation systems were developed for real-time monitoring of mission-critical systems such as power plants, water, gas and oil distribution systems. In recent years however, the practice of event correlation has gained greater application in network monitoring and in intrusion detection systems [63]. Network information that is typically logged includes: dates, times, port numbers, network addresses, hardware addresses, protocol types, and event IDs, their sources and their descriptions; this information typically plays a vital role within correlation efforts.

Kroon and Olivier [45] highlight that log entries are generated based on the purpose for the logging. For instance, if an administrator is only interested in statistical information, then it may only be necessary to monitor the services that members of the institution access. Furthermore, it may be acceptable to keep the data for short periods of time. On the other hand, if the logging is for security purposes, it may be pertinent to maintain audit logs of the institution members and the data being accessed. However, it may be necessary to maintain the log data for long periods of time.

4.5.2 Conditions for effective log correlation

According to Forte [26], all logs should have three fundamental characteristics in order to be effective for the purpose of correlation (and potentially for digital forensic investigations). These characteristics are *integrity*, *time stamping* and *normalization and data reduction*. These characteristics are now explored. Tudor [90] defines integrity as the protection of information, applications, systems and networks from intentional,

unauthorized, or accidental changes; this definition corresponds with Forte's interpretation. With this definition in mind, a system administrator must be certain that audit logs remain unaltered unless authorized changes must be made. Hashing is one of the mechanisms by which log file integrity can be maintained. However, Schneier and Kelsey [76, 77] state that no cryptographic mechanism can be used to actually prevent the deletion of log entries. Rather, this deletion can only be prevented through the use of write-only hardware such as writable CD-ROM disks, WORM (write once read many) disks, or paper printouts. Using the premise that no security measure can truly protect audit log entries after an attacker has gained control of an unsecured system, Schneier and Kelsey then present a protocol that upholds audit log integrity. The protocol ensures that, firstly, an attacker is unable to read log entries generated prior to the compromise of an insecure machine. Secondly, the protocol ensures that it is impossible for an attacker to 'undetectably' modify or destroy log entries.

Normalization, which is synonymous with event unification, refers to the ability of the correlation system to extract specific data from a log of a specific format. This is necessary for the purpose of correlation with data from log files of other formats, without affecting the integrity of the source log. Data reduction (or filtering) is necessary for the identification of pertinent events for the purpose of correlation, according to some predetermined selective criteria [26].

4.5.3 Maintaining log file integrity

Since log file integrity is typically 'attacked' after a system has been compromised, some information for protecting log files from such compromise is provided. Kroon and Olivier [45] discuss mechanisms by which the integrity and confidentiality of log files can be ensured. These mechanisms include authentication codes, firewall configurations, or a dedicated network. Authentication codes would ensure that other hosts are unable to spoof or falsify log messages. Firewall configurations can be set to restrict specific network communications in order to heighten security measures. A network dedicated to the task of logging can prevent attacks from directly 'hitting' the log servers. In addition, Kroon and Olivier propose an alternative method for

a logging system that is resilient to attacks — in the method, a silent log server and a dummy log server are used within a private network. The silent log server, or silent host, is able to receive network traffic, but due to an intentional hardware configuration, the host is unable to generate network traffic on the external interface to which it is connected. In their method, the dummy host is used for added security by hiding the fact that a silent logger is being used.

4.5.4 Legal considerations

Kenneally [42] offers a legal perspective for the use of logs within the jurisprudence of the United States of America. Kenneally begins by affirming that digital logs have become eyewitnesses. Kenneally further states that digital logs are not only used by technical administrators but also by business and legal professionals, according to the degree to which their work involves computers. With respect to the legal fraternity, two requirements must be met in order for logs to be admissible as evidence. These are *information assurance requirements* and *evidentiary legal requirements*. These requirements refer to the fact that the collection and storage of log evidence should occur in accordance with legal admissibility standards and must be in compliance with the specific investigation needs. In essence, digital evidence must be authentic, reliable and relevant to the case at hand. Fortunately, the admissibility of log evidence is heightened when an expert witness, familiar with the process by which the evidence is produced, testifies in support of any such evidence. Therefore within the legal arena, Kenneally's aim was to present information that would help minimize the number of successful challenges to the integrity of log evidence.

4.6 Chapter summary

This chapter provided a somewhat rudimentary overview on log files, log correlation and log file integrity. Nevertheless, the basis for the contribution was not unfounded — log files are the System Administrator's first incident response resource. Similarly, computer forensic specialists are able to glean a wealth of information through the

analysis and correlation of legitimate audit data.

The significance and usefulness of log data within investigations is often understated, although this source of forensic evidence almost always contains direct and subliminal inputs towards investigations. Therefore, the aim of this chapter was to place emphasis on the role of log files and how this can be leveraged within future forensic systems.

Chapter 5

Forensic Evidence Management System

5.1 Introduction

The vastness of digital evidence (and their sources) will always present a challenge within forensic investigations. For example, digital evidence includes, but is not limited to log files, network traffic and metadata (such as MAC times). Similarly, digital evidence sources are vast, including routers, application servers and local or removable storage devices [53]. From an investigation perspective, proprietary tools such as FTK [2] or EnCase [80] are utilized throughout the investigative process; Unix-based and open source tools such as **grep** and **dd** are equally utilized within the investigation process.

Given the multitude of digital evidence, digital evidence sources and forensic toolsets, an aggregated forensic evidence store would greatly assist investigators. It is upon this basis that we propose a design for a Forensic Evidence Management System (FEMS). The FEMS would provide investigators with the following:

- a holistic view regarding facts (evidence) within the investigation environment
- forensic evidence pertinent to the case at hand
- audits and insight into the quality of investigative inferences

The remainder of this chapter is structured as follows: we provide the underpinnings of the the Biba Integrity Model and Casey's Certainty Scale in section 5.2 and 5.3 respectively; the Biba model and Casey's scale are key inputs to the FEMS model. In section 5.4, we present the architecture of the FEMS and describe the components of the model. We discuss the flow of information within the model in section 5.6, where the value in the system is further highlighted. A chapter summary is then provided in sectionřefsummary forensic evidence management system

5.2 Biba Integrity Model

In an information security context it is widely understood that the goals of data integrity preservation are to:

- prevent unauthorized users from initiating modifications to data or programs,
- prevent authorized users from initiating intentional or unintentional modifications to data or programs, and
- maintain the internal and external consistency of data and programs

We therefore surmise that the integrity of digital evidence within the FEMS, and the protection thereof, is critical.

A possible tool to manage the integrity of data in the proposed evidence management system is the well-known Biba Integrity Model [66]. The Biba model is utilized to categorize evidential [digital] data into integrity classes or containers, where any application is only allowed to read data with a higher (or equal) integrity classification than its own. Furthermore, the application is only allowed to write data to containers with integrity classifications lower or equal to its own. Consequently, data can only flow from higher to lower integrity classes. Therefore, data of low integrity cannot contaminate data of higher integrity. Stated in a 'legal' context, evidence that has not been proven beyond reasonable doubt cannot contaminate evidence that already carries reliable evidentiary weight.

The Biba Integrity Model was derived by Kenneth J. Biba in 1977 and is the first model to address integrity within computer systems, based on a hierarchical lattice of integrity levels. The Biba model defines a set of access control rules, designed to maintain data integrity. Furthermore, the model was designed to address weaknesses within the Bell-LaPadula model — a preceding access control model which deals with confidentiality. Similarly to the Bell-LaPadula model, the Biba model makes use of *subjects(s)* and *objects(o)* that are ordered by an integrity classification scheme, denoted *I(s)* and *I(o)* respectively. It is through this classification scheme that object modifications are controlled within the model.

The Biba model is based on two fundamental properties, the *Simple Integrity Property* and the *∗Integrity Property*[1]. These properties are formally stated as follows [66]:

- **Simple Integrity Property**: subject s can *read* object o only if $I(s) \geq I(o)$. **(no read down)**
- **∗Integrity Property**: if subject s has read access to object o with integrity level $I(o)$, s can have *write* access to object p only if $I(o) \geq I(p)$. **(no write up)**

Simply stated, the Simple Integrity Property states that a subject at a given level of integrity may *not read* an object at a lower integrity level. The ∗Integrity Property states that a subject at a given level of integrity must *not write* to any object at a higher level of integrity.

Fundamentally, these rules are able to address untrustworthy information in a natural way. For example, let us assume that person A is known to be dishonest. If person A creates or modifies documents, then others who access and utilize this document should doubt the authenticity (or integrity) of the statements within the document. Furthermore, in a case where people are sceptical about a report based on flawed evidence, the low integrity of the report would naturally imply low integrity of any other evidence or deductions based on the report. Figure 5.1 provides an

[1] ∗Integrity Property is read as 'Star Integrity Property'

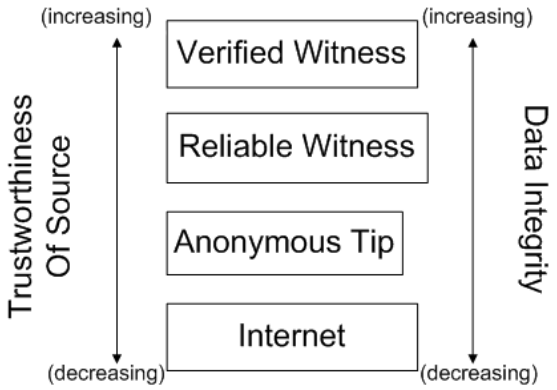

Figure 5.1: Trustworthiness of information, based on source

illustration of the trustworthiness, and therefore, integrity of information, based on the source of such information[2].

The notion of information trustworthiness, illustrated in Figure 5.1, is an all-encompassing portrayal of the manner in which the Biba model is incorporated within the FEMS.

5.3 Casey's Certainty Scale

Similar to other digital artifacts, log data within a network environment is susceptible to error. For instance, log data errors could result from log file tampering, data corruption or loss, lead-time in transmission, or simply an incorrect interpretation of the log content. Time-based differences (such as time-zone bias or differing system clock speeds) also contribute to the erroneous content or interpretation of such log evidence.

Understandably, errors within the forensic investigators core evidence base (that is log files) could render such evidence useless. Therefore, it stands to reason that levels of certainty must be associated with digital evidence, as well as with evidence sources. Casey's certainty scale [13] was developed to address the inherent uncertainties related to digital evidence in networked environments and we leverage this within the FEMS.

[2]Figure 5.1 was derived by the author, based on elements from an online source

In particular, Casey's certainty scale enables certainty (integrity) assessments to be associated with digital data. We illustrate Casey's proposal in Table 5.1.

Table 5.1: Casey's certainty scale [13]

Certainty Level	Description/Indicator	Qualification
C0	Evidence contradicts known facts.	Erroneous / Incorrect
C1	Evidence is highly questionable.	Highly Uncertain
C2	Only one source of evidence that is not protected against tampering	Somewhat Uncertain
C3	The source(s) of evidence are more difficult to tamper with but there is not enough evidence to support a firm conclusion or there are unexplained inconsistencies in the available evidence.	Possible
C4	Evidence is protected against tampering or multiple, independent sources of evidence agree but evidence is not protected against tampering.	Probable
C5	Agreement of evidence from multiple,independent sources that are protected against tampering. However small uncertainties exist(e.g., temporal error, data loss).	Almost Certain
C6	The evidence is tamper proof and unquestionable.	Certain

The first column of Table 5.1 lists the seven certainty (integrity) levels, from the lowest(C0) to the highest(C6). The second column indicates the preconditions leading to the integrity conclusions in the third column. Note that the higher the certainty (integrity) level, the greater the integrity associated with the evidence source and, hence, the inferences based on the evidence.

A prerequisite for the incorporation of the Biba Integrity Model into the FEMS is the establishment of an integrity classification scheme; we adopt Casey's Certainty Scale [13] for this purpose. Casey's certainty scale provides a mechanism to associate

certainty to specific facts [electronic data] or inferences within a networked environment. Since certainty equates to integrity, we are able to extend the application of Casey's certainty scale to the FEMS. This forms an ideal starting point for describing the FEMS that is expected to manage digital evidence with cognisance of the integrity of such evidence. Henceforth the terms certainty and integrity are used interchangeably.

Sections 5.2 and 5.3 provide the core inputs to the proposed Forensic Evidence Management System (FEMS). In addition, the rule-based correlation technique (presented in section 4.3) is embodied within the FEMS; the fundamental elements of this correlation technique, such as its simplicity, intuitive rule generation and the derivation of inferences are essential and requisite to the FEMS description in this work.

In the following section we illustrate and describe the FEMS architecture. Thereafter, with real-world and digital-world scenarios, we undertake a preliminary discussion into the information flow within the FEMS.

5.4 Forensic Evidence Management System Architecture

The FEMS, as illustrated in Figure 5.2, is constructed using a component-based architecture. That is, the various system components are distributed within a client layer, a logic layer and a data layer. Hence, the system is designed using a classical business system model. Similarly to a business system, the client layer serves as the investigator's interface into the system. The logic layer houses the processing rules for the system and the data layer stores the raw data which an investigator indirectly interacts with.

5.4.1 Client layer component

The *system interface* is the channel through which forensic investigators are able to access facts within the FEMS. The investigator uses queries to interrogate the system

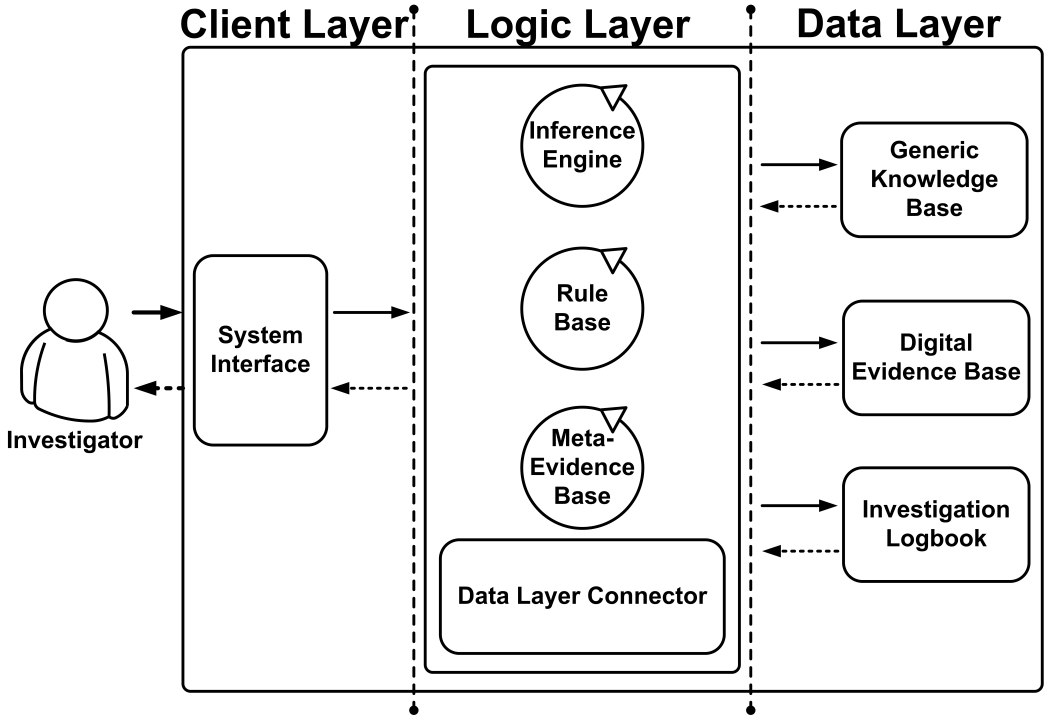

Figure 5.2: Forensic Evidence Management System (FEMS) Architecture

for evidence; these queries include hypotheses that the investigator tests against facts within the system. The system interface also enables an investigator to update data within the system.

5.4.2 Logic layer components

The *rule base* is a store for the action(s) to be taken, given any particular fact within the system. The rule base also represents knowledge about facts that can be inferred from the system [10]. We assume that rules within the system may not necessarily exist *a priori*. However, they may be entered *a posteriori*, based on the specific case at hand — this assumption is explored further within section 5.5. In addition to the derivation of inferences within the proposed system, a rule base is incorporated due to its ease of implementation and the intuitive nature of rule generation [63]. For instance a rule may specify an integrity label to be associated with facts from, or derived from a specific source of forensic evidence.

The *meta-evidence base* is directly interfaced with the rule base and specifically houses only inferred evidence. Therefore, queries to the system that have been confirmed or refuted within the system are routed and noted here. The *inference engine* is where Casey's certainty scale is implemented. This component draws inputs from the rule base and the meta-evidence base in ascribing certainty labels to inferences and evidence within the system.

Software engineering best practice requires that a system user should never directly interact with a system's data source. This forms the basis for the incorporation of a *data layer connector*, thereby providing the necessary abstraction between the investigator and the raw evidence within the system.

5.4.3 Data layer components

The automated documentation of actions within investigations is certainly not new. Such functionality exists within forensic tool suites such as EnCase [80], FTK [2] and ProDiscover [65], where information such as technician name, project name and description and the date and time of specific actions are recorded; in the case of FTK, these reports are even customizable. Similarly, the *investigation logbook* is used to record all investigative actions performed within the system. However, in the context of the FEMS, investigative steps such as system queries or hypotheses, rules or data that have been added and inferences from the system are noted. Therefore, the value of this is in providing a trace of all system inputs and motivations that an investigator uses in reaching investigative conclusions (to prove or disprove a hypothesis).

The *digital evidence base* is the interface to classic forensic evidence sources such as the differing forensic tool suites (Forensic Tool Kit or EnCase). Also incorporated into this evidence base would be inputs from log correlation sources, logical access records (including access control matrices) and physical access records such as work attendance registers. The *generic knowledge base* is to this system what fingerprints are to the physical world. This knowledge base houses static information such as software names, versions and descriptions. This also includes a database of known files such as standard operating system files or hashes for generic file signatures such

as .gif or .jpg encoded files. Within the system this component functions (and is referenced) in the same way as the Known File Filter within FTK and the NIST maintained National Software Reference Library [62].

In the following section we present a preliminary discussion on information flow within the FEMS. Further details pertaining to the architectural components are also provided.

5.5 Preliminary discussion

Digital evidence in the FEMS is stored as facts. For example, a fact within the FEMS would be:

1. *web proxy 223.1.3.27 return code 200 = O.K: Request succeeded*
2. *web proxy 223.1.3.27 return code 400 = Bad request: request could not be understood by the server*

Fact 1 is interpreted as, *"client request successful"*, while fact 2 is interpreted as, *"client request not understood by the proxy server"*.

Without loss of generality, we proposed that such facts are stored in the form of propositions or predicates. For example, for a murder case in the physical world, the fact that gunshot residue (GSR) was found on a suspect (S) may be represented by the *GSR(S)* predicate. Similarly, if the suspect has been placed at the scene of the crime, the corresponding predicate is *AtScene(S)*. Other predicates may be used to represent various facts about S, about the crime itself, and about other 'agents' involved in the crime.

At any point it is possible to deduce new facts, based on deduction rules. As mentioned earlier, the rule base specifies actions to be taken, given any fact within the FEMS, or could represent knowledge about facts that can be inferred. Such rules may be applied in an automated fashion, in which case the use of deductive databases may be ideal. Alternatively, deductions may be made by human investigators, where rules may be entered to expound on and track investigative logic. The precise nature

of such deductive rules and their effectiveness are not within the scope of this work and are not discussed further.

We surmise that it is possible to express logic using such rules as is the case in existing systems, such as the system described by Burns et al [10]. We differ from such systems in our outlook, that not all rules will exist in the system *a priori*, but that they may be entered *a posteriori*, based on the specific case at hand. We also assume that inferences may be made in a forward or backward manner — that is, the system may derive all possible facts, or a given hypothesis may be tested against the known facts.

At this juncture we recall the fundamental differences between the physical world and the digital world. The distinction between these realms is important in the current discussion because, unlike the physical world, it is possible to automatically derive facts in a digital world from augmented forensic tools. For example, suppose that it is relevant whether a certain picture, or certain words in an email, occurs on a suspect computer. Consider that such a disk has been imaged; forensic tools that are utilized for analyzing the disk will be able to automatically add such facts to an evidence knowledge base. However, we note that not all facts in a digital world can be derived automatically — for instance, the question of whether or not a picture found on a suspect's computer is a pornographic image may only be answered by a human investigator.

This discussion has not yet addressed integrity or Casey's certainty scale. To consider these, it is necessary to label facts within the evidence management system with an integrity label. For instance, if an image (I) occurs on a disk, it may be represented as *OnDisk(I, C6)* [3]. Furthermore, certain deduction rules may be applicable within the system, irrespective of the integrity of the facts they operate on. For example, suppose it is of interest whether an image containing embedded data — hidden by means of steganography — exists on a machine; suppose that it is also of interest whether tools to decode such a message are present on the machine. In other words, an investigator would seek to determine whether evidence exists that it was

[3]Henceforth, references to Casey's certainty scale are denoted *Cx*, where x represents the certainty label number.

possible to read the message; this rule will clearly apply to whatever the certainty of its preconditions is. Alternatively, a rule that simply categorizes the image, based on the applied compression algorithm, would also be applicable regardless of the image integrity label.

In line with the Biba model, the certainty of a fact derived by such a rule will depend on the certainty of its preconditions. In actuality, a new fact will, in general, have the lowest certainty of its preconditions. For example, suppose that an image is retrieved from a removable disk which has been corrupted through a virus infection. Understandably, this source cannot be entirely trusted. However, new facts that are logically derived will not always have the lowest integrity of its preconditions; the Casey scale states that the certainty of a fact increases if that fact is supported by independent sources. Let us consider the requirement to be classified on level *C5*: *"Agreement of evidence from multiple, independent sources that are protected against tampering. However small uncertainties exist (e.g., temporal error, data loss)."*. Therefore the converse to the earlier example would be as follows: if the image was retrieved from a secure site or secure FTP application, which is then corroborated through the site or FTP session logs within the evidence management system, then the integrity of the image and other evidence based on the image is improved. In other cases some facts that have been well established, may lead one to form a rather tentative hypothesis about some other facts, in which case the certainty of the new fact will be lower than its preconditions. Consequently, there is a need for trusted upgraders and downgraders — this concept is discussed in the following section.

In the following section we elaborate further on the flow of information within the FEMS. This elaboration is achieved by means of a computer intrusion scenario, the nature of which warrants a thorough investigation. The concept of upgraders and downgraders is also explained.

5.6 Information flow within FEMS

Consider an intrusion that exploits a web browser vulnerability. We further assume that, by harvesting authentication information from the compromised computer, the intruder accesses a financial system. Given that the FEMS is accurately populated, the forensic investigator would interrogate FEMS for configuration files, source code, executable programs (such as rootkits), Internet activity logs, or password protected text files. These queries would be submitted using the FEMS interface and then brokered by the data layer connector, which parses information returned by the data layer.

Suppose that the intruder successfully modifies event logs during the attack. Therefore, the Internet activity logs may have been tampered with. However, if these logs had been generated and sent directly to a secure log correlation server, then it may be inferred, and set within the rule base that *LCorrelation(Internet log, C6)*. That is, the log-related information is tamper proof and unquestionable.

At this point, the intruder's access must be verified in the audit logs of the financial system. However, assume that the financial system's audit logs are deemed to be unreliable because they are not explicitly protected against tampering; this situation can be expressed by the fact *FinSyst(log, C2)*. Using this fact and the inference rule: *for (LCorrelation(log, C6)* $\geq$ *FinSyst(log, C2)) update_meta-evidence*, it would be deduced within the inference engine (and sent to the meta-evidence base) that, although the financial system logs verify that the 'victim's' credentials were used at a specific time, conclusions based on this information may not be trusted per se.

We are now in a position to discuss the concepts of upgraders and downgraders. An upgrader is any evidence or evidence source with an integrity label $\geq$ C5 (and corroborated by two or more trusted evidence sources), which is used to improve the certainty associated with facts or inferences within the FEMS. Table 5.2 presents a sample upgrader matrix.

In contrast, a downgrader is any evidence of evidence source with an integrity label $\leq$ C2. Table 5.3 presents a sample downgrader matrix.

Upgraders and downgraders are influential because they cause the inference engine

Table 5.2: Upgrader matrix

	C0	C1	C2	C3	C4	C5	C6
C0	C0	C0	C0	C0	C0	C1	C1
C1		C1	C1	C1	C1	C2	C2
C2			C2	C2	C2	C3	C3
C3				C3	C3	C4	C4
C4					C4	C5	C5
C5						C6	C6
C6							C6

Table 5.3: Downgrader matrix

	C0	C1	C2	C3	C4	C5	C6
C0	C0	C0	C0				
C1	C0	C0	C0				
C2	C1	C1	C1				
C3	C2	C2	C2				
C4	C3	C3	C3				
C5	C4	C4	C4				
C6	C5	C5	C5				

to modify evidence integrity labels. Therefore, in this example, the log correlation evidence source is considered to be an upgrader. This is because, all else equal, the implementation of a correlation solution is typically fortified. Furthermore, as a direct consequence of the Biba model properties, the log correlation evidence source is allowed to upgrade the integrity label of the financial system log. Therefore, using the matrix presented in Table 5.2, the inference engine upgrades the integrity label of the financial system log to C3.

Although the financial system logs may be included within the log correlation system, they may not positively influence the integrity of other evidence within the system until their own integrity is enhanced. Throughout this process, the investigation logbook is programmatically instructed to record all logical steps and inferences.

5.7 Chapter summary

The preceding chapters have laid the foundation upon which the Forensic Evidence Management System (FEMS) is based. In chapter 2 we provided an overview of a number of traditional Internet-based crimes. In chapter 3 we dealt with the computer forensic process and the mechanisms utilized in conducting digital investigations. Chapter 4 underscored the need and importance of log files, log file integrity and log correlation within forensic investigations.

In this chapter we introduced the model for a Forensic Evidence Management System (FEMS), intended to assess, determine, preserve and subsequently reason about the integrity of digital evidence during forensic investigations. The solution employs the well-known Biba Integrity Model to manage and maintain the integrity of digital evidence hosted within the system. In conjunction, Casey's Certainty Scale is selected as the integrity classification scheme for the application of the Biba model.

The FEMS architecture incorporates a rule base, a meta-evidence base, an inference engine, a digital evidence base and a generic knowledge base for reasoning about the integrity of evidence; the architecture also offers a system interface for evidence input and queries. The investigation logbook is incorporated to record all investigative actions, thereby providing an audit trail of all investigative deductions.

From a legal perspective, investigative deductions based on erroneous or questionable evidence is arguably more damaging than the lack of evidence within a case. Therefore, the principal benefit of FEMS is that, when provided all the relevant input sources, it provides investigators with holistic views of the forensic evidence pertaining to a case and insights into the quality of their investigative inferences. The FEMS also provides a novel mechanism for managing the integrity of digital evidence within networked environments, in spite of the inherent irregularities associated with digital evidence within such environments.

Chapter 6

Cyber Crime Profiling

6.1 Introduction

The field of digital forensics is arguably within an adolescent stage, where current shortcomings within legal texts, investigative processes and forensic tool-sets are generally well known. However, resolution to these known shortcomings, to date, have been inadequate. From a technical and administrative perspective, forensic specialists are unable to keep pace with the rate of technological growth, coupled with the forging of next-generation networks. Understandably, the tools commissioned during investigations and the applicable laws within legal jurisdictions may always lag behind this growth. Furthermore, the growth of the Internet is an enabler to numerous business and social applications. However, this rise in applications has ushered in a new breed of criminals, namely cyber criminals, who expose and exploit vulnerabilities within operating systems, applications and networks connected to the Internet.

The fore-mentioned challenges were depicted in a case against Gary McKinnon, commonly known as the NASA hacker [96]. Gary McKinnon (a Briton computer systems administrator) was accused of penetrating 97 United States military and NASA computers in 2001 and 2002. It was also confirmed that the affected network hosts were not password protected; McKinnon has explained that he was able to access these networks simply by using a Perl script that searched for blank passwords. In 2002 McKinnon was arrested by the UK National Hi-Tech Crime Unit under the

Computer Misuse Act, and at the time, he was informed that he would face community service. Later that year, McKinnon was formally accused by the US government of having committed a criminal offense. The US government has further requested that McKinnon be extradited and tried in the US; he stands to face up to 70 years imprisonment if extradited. In this context we note that the laws within legal jurisdictions do differ. Notably, the penalties within these legal jurisdictions also differ significantly.

Although some legal considerations have been mentioned thus far, these are not within the scope of this work. Rather, in this chapter we continue to describe the components of the integrity-aware Forensic Evidence Management System (FEMS). We make use of a finite state automaton (FSA) to model the FEMS's behaviour, and in so doing we demonstrate how cyber crime profiling can be achieved.

The remainder of this chapter is structured as follows: in section 6.2 an overview of the different types of (digital) trace evidence is provided. Thereafter, the trace evidence sought out during certain cyber crimes is provided; these inputs are used towards the cyber crime profiling exercise, where trace evidence is mapped to specific cyber crimes. The use of a finite state automation towards cyber crime profiling is described in section 6.3. Specifically, the components and transitions within the automaton are depicted and used to demonstrate the FEMS's behaviour. Elementary theory regarding finite state automata is also provided in this section. In section 6.4 and section 6.5 we demonstrate the behaviour of the FEMS FSA with a walk-through of a child exploitation and a computer intrusion investigation scenario respectively. The chapter is then concluded in section 6.6.

6.2 The nature of digital evidence

In reasoning about the workings of the FEMS, we begin with a review of the types of digital evidence sought out during investigations. Two categories of cyber crimes are then provided with a mapping between these crimes and the evidence typically used to prosecute on such crimes.

6.2.1 Types of digital evidence

Digital evidence can be categorized into user-created files, user-protected files and computer or system created files. Due to the self-explanatory nature of these categories we only provide examples of such files. Examples of user-created files are address books, email files, audio and video files, image files, calendars, Internet bookmarks, database files, spreadsheet files and document files. Examples of user-protected files are compressed files, misnamed files, encrypted files (for instance, encrypted data vaults), password-protected files, hidden files and steganographically manipulated files. User-protected files are often harder to identify and 'interrogate'. This is because users are able to employ advanced tools, with relative ease, in encrypting or masking trace evidence. In certain instances users deliberately rename incriminating files to seemingly harmless names to avoid detection. Examples of system created files are backup files, log files, configuration files, printer spool files, cookies, swap files, hidden files, system files, history files (especially Internet history files) and temporary files [91].

6.2.2 Mapping digital evidence to cyber crimes

The cyber crimes of child exploitation (or abuse) and computer intrusion are used in achieving the desired mappings — these are examples of e-enabled and true cyber crimes respectively. This choice of crimes also illustrates the varied contexts within which digital media is typically interrogated. That is, within a child exploitation case, the digital evidence would largely reside on the perpetrator's local disk, or on removable storage media. Furthermore, evidence of such a crime also typically reside within Internet activity logs. However, within a computer intrusion case, the digital evidence is dispersed due to the networked nature of this crime — the source of the attack may be the perpetrator's computer, while the target may be a host computer within a different geographical location and the trace evidence may not necessarily reside on a single path between these network hosts.

In a child exploitation case, investigators typically identify the following information: images, emails, video files, Internet chat logs, Internet activity logs, digital

camera software, user-created directories,and graphic editing and viewing software. In a computer intrusion case an investigator would identify the following information: emails, configuration files, Internet chat logs, Internet activity logs, source code, executable programs, text files (with user names and password) and a network address and user name [91]. The similarities between the evidence gathered within these crime investigations is quite clear. It should however be noted that specific evidence distinguishes these crimes. For instance, the evidence of source code or executable files (such as root kits) would support a hypothesis that the perpetrator was engaged in a computer intrusion.

This preliminary discussion provides an ideal backdrop to introduce a finite state automaton in illustrating cyber crime profiling. This is explored further in the following section.

6.3 Cyber crime profiling

Using the concept of a finite state automaton (FSA) [52, 95], the crime scenarios above are used in modeling the FEMS's behaviour. In so doing, transitions within the FSA illustrate paths towards a cyber crime profile. We provide preliminary concepts and notation for the FSA in the following section. Thereafter, the FSA is illustrated and an explanation of the states and transitions within the automaton are provided. The section is then concluded with an explanation of the FSA's behaviour within a child exploitation scenario and a computer intrusion scenario.

6.3.1 Finite state automata: concepts and notation

Some brief concepts around finite state automata are provided as follows: Each state stores information about the past; that is, it reflects the input changes from the system start to the present moment. A transition indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An action is a description of an activity that is to be performed at a given moment [95]. However, there are several action types; without loss of generality, each action within

the FSA will represent a transition action type. A formal definition for a FSA can also be found in [52].

The following predicates are defined in the FSA (based on notation used in section 5.2): S_i will represent evidence sources (or subjects), E_i will represent digital evidence (or objects) and $I(\text{x})$ will represent certainty values assigned to subjects or objects within the framework. That is, x can either be S_i or E_i. We also define R_i to represent rules within the rule base. In all instances $i \in \mathbb{N}$.

6.3.2 FSA states and transitions

In this subsection we provide an illustration of the FSA in Figure 6.1. Thereafter, we describe the states and transitions of the automaton. Furthermore, we provide the rationale of each state and its transitions within the automaton.

The FSA consists of four core states, namely the hypothesis, decision, rule and data states. These states are mapped to the respective components within the FEMS; that is, the System Interface, the Inference Engine, the Rule Base, the Generic Knowledge Base and the Digital Evidence Base respectively. The transitions within the FSA are as follows:

- ENTER FACT(E_i, S_i, $I(\text{x})$)
- EXECUTE RULE(E_i, S_i, $I(\text{x})$)
- UPDATE RULE(R_i)
- GENERATE INFERENCE(E_i, S_i)
- INFERENCE RESULT(E_i, S_i, $I(\text{x})$)
- MODIFY CERTAINTY(INFERENCE RESULT(E_i, S_i, $I(\text{x})$))
- REQUEST(E_i, S_i, $I(\text{x})$)
- REQUEST RESPONSE(E_i, S_i, $I(\text{x})$)

As depicted in Figure 6.1, the hypothesis state acts as the start and end state of the automaton. This is because initial hypotheses and final outputs are entered and returned to this state respectively. It may also be required that a fact be amended within an investigation; this is also achieved in the hypothesis state.

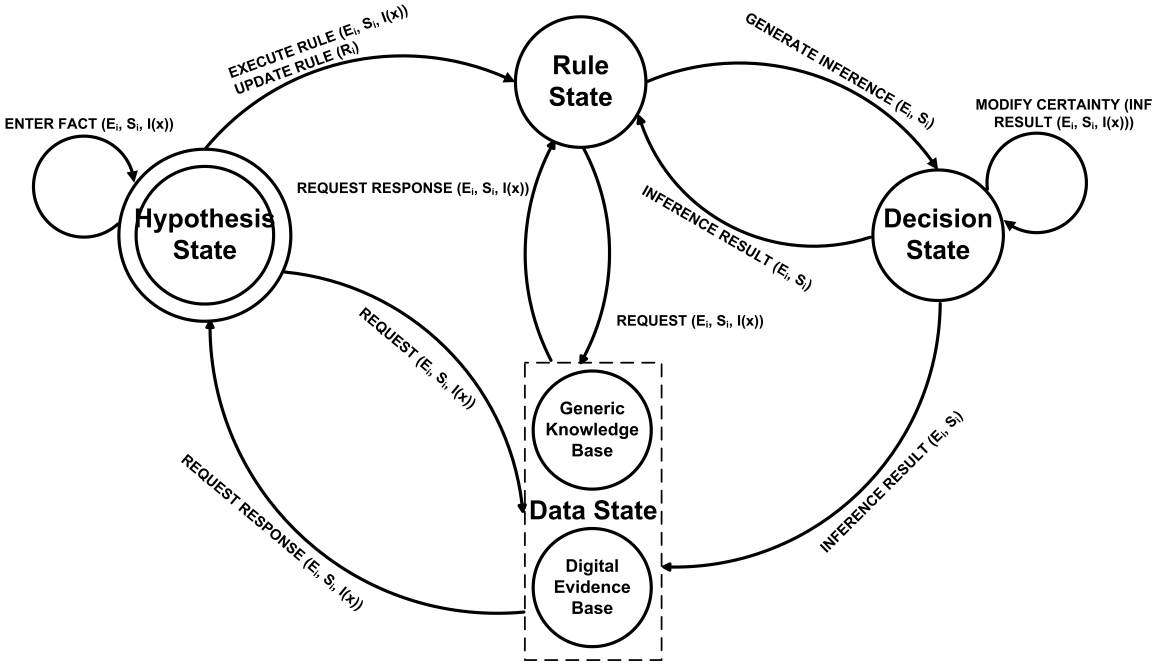

Figure 6.1: Finite State Automaton depicting the FEMS's behaviour

The rule state is entered when an investigator requires a rule to be executed on input data, or would like to amend a rule R_i. Based on the applied rule, actions are either triggered to the decision state, the data state, or both. The interaction between the rule and data states is a significant one. Certain rules may simply request information from the data state, while other rules may specify amendments to be made within the data state. The cyber crime profile forms the core of the rule base. That is, the cyber crime profile is encapsulated within this state. Therefore, a specific sequence of rules are defined and must be executed before the crime profile is realized. The scenarios provided in sections 6.4 and 6.5 expand on this further.

Once a rule is applied, an inference action is triggered. The inference action is executed within the decision state, where the certainty value of the evidence or

source in question is updated accordingly. Where applicable, an inference result may be required to update a rule(s) within the rule state. For example, if the certainty associated with a log file is degraded by the fact that the log file has been tampered with, then rules applicable to the log file should be amended accordingly.

A significant action within the decision state is the *MODIFY CERTAINTY* action. This action ensures that all certainties within the inference base are updated whenever an inference result is generated. This action is iterative to ensure that the influence of any inference result on system objects is known at all times. Similarly to the rule state, there are instances where inference results may require an amendment to data. Using the example of a tampered log file, the hash value of the log file may be inconsistent with the hash value within the generic knowledge base; this would certainly need to be amended.

The data layer connector and the meta-evidence base were intentionally omitted within this elaboration. This was done in an effort to reduce overall complexity and transactions within the automaton. For this reason, the *REQUEST* and *REQUEST RESPONSE* actions are depicted directly between the hypothesis and data states.

In summary, an investigation would typically begin where an investigator enters an initial hypothesis or fact into the system. Based on this hypothesis, all evidence within the system would be traversed until a decision is determined in a heuristic manner. Whenever tentative inferences are made, the decision state would be responsible to amend the certainty values of all evidence and sources accordingly. Therefore, the final system output is provided only where the certainty assigned to output parameters are at an optimum. Based on this optimum, an investigator would verify the system output. That is, if the certainty assigned to system outputs is above C4 or below C5, then the investigator would make an 'accept' or 'reject' decision respectively.

The 'real world' scenarios of child exploitation and computer intrusion are presented in the following sections. In so doing, the considerations for cyber crime profiling are further presented. For simplicity, we assume that the crimes in question are perpetrated within a managed network environment.

6.4 Child exploitation scenario

Child exploitation cases are arguably simpler to prove than other cyber crimes. This is largely because, from a real-world perspective, the psychological profiles of such offenders are generally well known. For instance, perpetrators conduct such crimes in a covert manner, and their digital traces are often concealed accordingly.

Similarly to other investigations, an investigation of such a crime would begin with an initial hypothesis. For example, an initial hypothesis could be that person A is the perpetrator of the cyber crime. This hypothesis would be provided to the hypothesis state, and information such as person A's user name and password would initially be entered into the system as evidence. Ideally, the suspect's other system or network application credentials would also be supplied as initial facts. This is acceptable since such preliminary information is typically subpoenaed from suspects. An initial certainty value of C4 can be assigned to this information, indicating a reasonable degree of certainty.

From the hypothesis state, an applicable rule is executed on the evidence provided. Let us consider that an initial rule applied to all facts is a verification step. A request with all the relevant input parameters would then be sent to the data state. The password hashes of the supplied credentials would be compared to those within the data state (specifically within the digital evidence base). A response would then be parsed to the requesting rule. Based on the data state response, an inference generating action would be required from the decision state — this is achieved through the generate inference action. Therefore, if the data state response reveals that the hashes are identical, then firstly, the certainty associated with the initial evidence can be elevated to C5 (the second highest level of certainty associated to evidence). Secondly, the elevated certainty value is substantiated in that an identity management store (which is an authoritative source for network credentials) verified the password hashes.

The notion of memory is encapsulated within each state, further motivating the use of a FSA. However, this feature is predominantly realized within the decision state. As depicted in Figure 6.1, a fundamental and continuous action within this

state is the modification of certainty values within the FEMS. This update action is performed whenever an inference result is derived. Furthermore, the inference result is used as input for such update actions. Therefore, based on the assertion above, the certainty value of subsequent inferences based on the suspects (verified) credentials are likely to be elevated.

The discussion above has centered around an initial pass through the FSA, where an initial hypothesis has been entered and a verification rule is applied. The next pass would execute the next rule in the sequence of rules defined by the crime profile in the rule state. For example, the next rule could specify that the suspect's computer be searched for image files. However, due to the nature of this crime, the search results would require human interpretation. Based on the investigator's interpretations, new certainty values are assigned to the discovered evidence. A frequent occurrence during investigations is the discovery of images within Internet history folders. At this stage, the data state could be further queried with regards to person A's Internet activity logs. The certainty assigned to this inference is elevated if the Internet activity logs are corroborated with the images found within person A's computer Internet history logs. Subsequent rules towards identifying this cyber crime could be an interrogation of the computer for Internet chat logs, digital camera software and password-protected or encrypted files. Using augmented forensic tools, a rule could be specified to interrogate ambient spaces (for deleted but existing files) on the suspects computer — this would include unallocated space, file slack and swap files.

It is also not uncommon for investigators to discover new facts during an investigation; this is part and parcel of the investigation process. In certain instances, new evidence can dramatically change the course of an investigation. Furthermore, new evidence potentially influences the workings of states such as the rule base. For this reason, the FEMS also provides functionality to enter new evidence and to make rule amendments on an ad-hoc basis. Consider a scenario where, at a late stage in the investigation, it is discovered that person A was not actually within the organization when the crime was perpetrated; perhaps he or she was on vacation. Such a discovery would off-set a number of the deductions described thus far.

The child exploitation scenario is a gentle reminder of the dynamic nature of

investigations; it also provides a glimpse into the challenges within more intricate cyber crimes, thereby reinforcing the importance of the interrogation of multiple evidence sources before investigative conclusions are reached. In the following section we briefly present the workings of the FEMS and the FSA using a computer intrusion scenario.

6.5 Computer intrusion scenario

The identification of perpetrators in cases of computer intrusions is typically difficult to detect. This is largely due to the obfuscation provided by the Internet and certain security technologies. For instance, proxy serves are able to provide network address translation features, which inherently obscures the source of an Internet-based request.

In general, attackers are able to commit this crime by exploiting system weaknesses, or compromising legitimate system user credentials and masquerading as such. The initial rules and rationale for detecting this cyber crime with the FSA are similar to the explanations provided in section 6.4. Therefore, only distinguishing factors in this cyber crime is provided further.

Given the global standardization of network identity, through the use of the TCP/IP protocol [46, 67, 68], an initial hypothesis for such a crime could be that the source address of the attack is *10.51.184.34*. The nature of this crime would further require that crime profile rules interrogate perimeter security technology logs, namely Intrusion Prevention System logs and Firewall logs. The certainty value assigned to these technologies is C5, since these technologies are used to fortify demilitarized zones. Based on the source network address, a subsequent rule could be for a *traceroute* command to be applied on the address — this would provide further insight toward the source of the attack.

From a private network perspective, rules and inferences are also applied on the target computer in determining the existence of malicious content such as rootkits, source code and other unexpected executable files.

6.6 Chapter summary

The obfuscated nature of the Internet, the lack of standardization in laws within differing legal jurisdictions, the lack of event correlation, the widening search area and the lack of trained forensic specialists are all significant challenges within digital forensic investigations. In this chapter we expounded on the operation of the forensic evidence management system (FEMS) by making use of a finite state automaton (FSA) to develop and reason around the FEMS's behaviour.

In particular, we described how sample rules within the rule state of the FSA could be crafted to recognize and profile cyber crimes. The digital evidence typically used to confirm or refute investigative hypotheses was also consistently detailed. The demonstration of the FEMS's operations, by means of cyber crime profiling scenarios was by no means exhaustive. However, the scenarios provided vital input toward cyber crime profiling.

Chapter 7

FEMS Processing Algorithms

7.1 Introduction

The value in the Forensic Evidence Management System (FEMS) lies in its ability to provide forensic investigators with a holistic view of an investigation and to contribute positively towards profiling the source(s) of incidents, thereby honing search activities within investigations. Therefore, the FEMS would aid in the efficient allocation and utilization of (limited) investigative resources — whether human, software, or in investigative instrumentation.

In the preceding chapter, the FEMS finite state automaton (FSA) is provided to illustrate the FEMS's general behaviour, where the states and transitions within the FEMS FSA represent the general interactions between the FEMS's core components.

In this chapter we expand on the core states described within the FSA — we develop processing algorithms for the hypothesis state, rule state, decision state and the data state within the FSA. This elaboration is achieved through the use of flowcharts, depicting the following:

- processing steps within the said FSA states
- input and output parameters for the transitions, and
- the decision points influencing the probative value of the inferences within the FEMS

In developing these flowcharts, we're able to establish fundamental algorithms for processing of information by components within the FEMS.

The remainder of this chapter is structured as follows: in section 7.2 we provide a set of assumptions under which the elaborations are based. In section 7.3, flowcharts for the hypothesis process, rule process, decision process and data process are provided. Thereafter, we discuss the individual flowcharts, with emphasis on the decision points and information flow control within these flowcharts. Section 7.4 provides the processing algorithms for the states within the FSA; these algorithms are extrapolated from the flowcharts generated in section 7.3 and depicted in pseudo-code. A chapter summary is then provided in section 7.5.

7.2 Assumptions

In order to effectively demonstrate the processing algorithms and component interactions within the hypothesis, rule, decision and data phases of the FEMS, the following assumptions are maintained for the remainder of this chapter:

- The FEMS is applied within the context of a managed network environment, where all network components are known (to the FEMS) and are capable of generating and storing log evidence
- The application of the FEMS is extendible to a number of environments, one of which is the Internet. However, due to the potential complexity of an analysis exercise, the application of the FEMS is limited in this work. For instance, within the context of the Internet, the FEMS would need to consider a number of evidence sources and the analysis would need to incorporate a multiplicity of factors inherent to the Internet environment
- We predominantly consider the analysis phase within an investigation. In so doing, we assume the pre-existence of evidence such as data integrity checksums, images of source evidence and even log files, all of which are evidentiary artifacts collected prior to the analysis phase of an investigation

On the whole, these assumptions enable us to provide the processing algorithms, unconfined by the inherent details contained within a network environment.

7.3 FEMS component processing flowcharts

We now develop on the hypothesis state, rule state, decision state and data state provided in Figure 6.1. We make use of flowcharts to depict the flow of information and component interactions within these states, thereby providing an outline of the algorithms for these FEMS components.

In each of the subsequent figures in this section, a distinction is drawn between the manual activities within an investigation and the FEMS processing activities. Furthermore, the interaction between manual and automated activities within the model illustrates the necessity for human intervention and/or interpretation within any forensic investigation; in the author's opinion, certain human tasks cannot be discounted from investigations, even within an automated investigative system.

7.3.1 Hypothesis state flowchart

The hypothesis process flowchart is depicted in Figure 7.1. This flowchart has four distinct components: the initial decision phase, the information processing phase, the information update phase and the final decision phase.

The intentions of an investigator are established within the initial decision phase — that is, the system prompts the investigator on whether (s)he needs to retrieve information stored within the system, modify or enter facts (evidence) into the system, or to modify or create rules to be effected on evidence within the system (during the course of the investigation).

The processes within the information processing phase are executed after the investigator's initial actions are determined within the initial decision phase. As a result, auxiliary tasks within this phase may interrogate, retrieve, or update evidence within the data layer of the system. For example, in the instance where the investigative decision is to only retrieve information, the *Request Response* process would

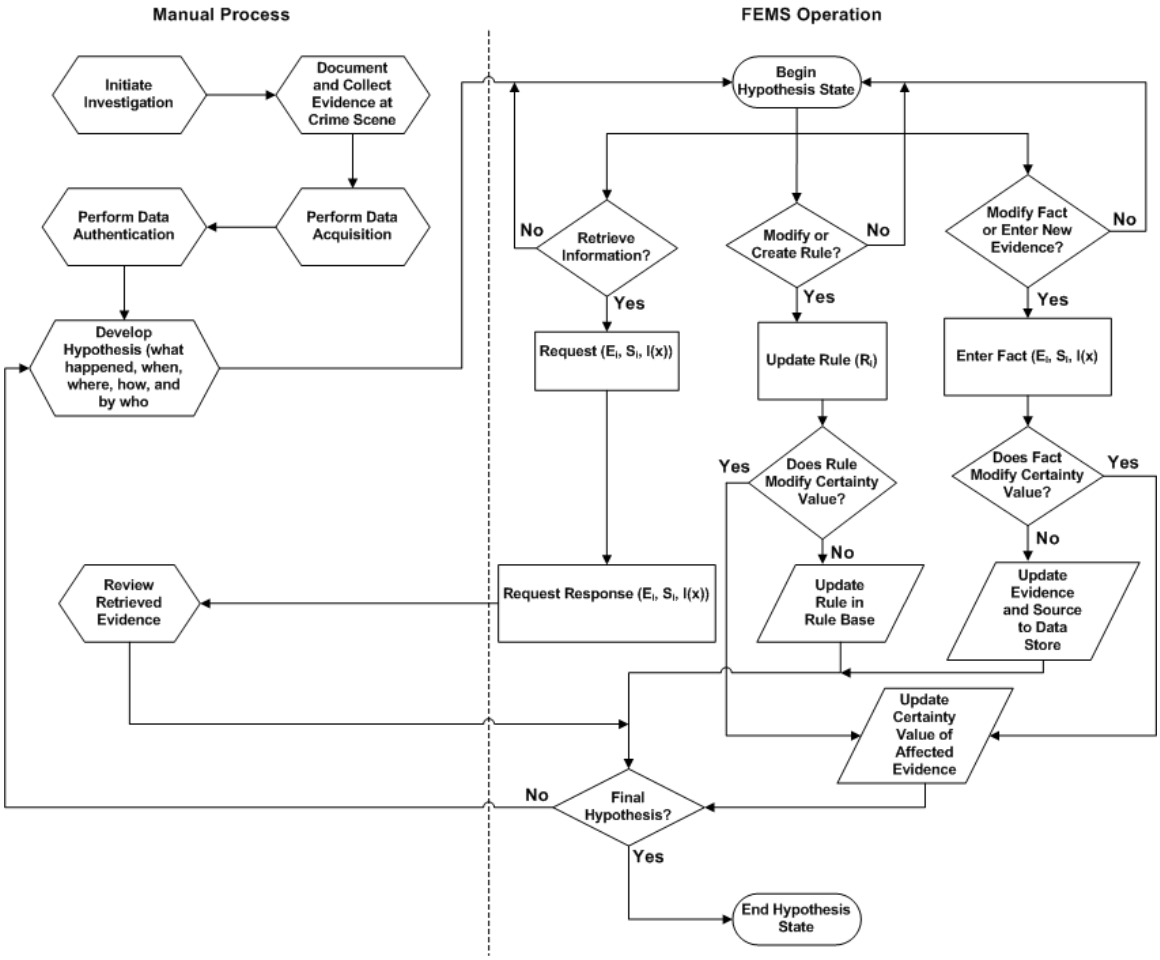

Figure 7.1: Flowchart for the Hypothesis process

trigger an auxiliary task to the appropriate data layer component.

In the information update phase, the appropriate data components within the FEMS are updated with the investigators decisions and or new information.

The final decision phase is initiated subsequent to the low-level activities resulting from the information processing phase, as depicted in Figure 7.1, whenever data is updated to the data layer, or returned to the system's user, a final decision is made on whether another iteration of the hypothesis phase is required by the system user.

It should be noted here – and in the following subsections – that references to the "update" of information or evidence within the flowcharts does not refer to the

tampering of digital evidence. In this context, the word "update" is used to refer to information transformations within the FEMS's storage mechanisms, thereby enabling the system functionalities. For example, amendments to the integrity associated to evidence within the FEMS are deemed as amendments to meta-data within the FEMS's data store.

7.3.2 Rule state flowchart

As suggested in Figure 7.2, the rule process flowchart is typically activated within the data analysis phase of an investigation. The process begins where the system collates all the rules to be effected on the evidence within the FEMS, based on the investigative hypothesis at hand; this approach is consistent with the manner in which manual investigations are conducted, especially since there are specific considerations and evidence stores that are interrogated throughout the analysis cycle.

The sequential execution of the rule-set commences after the rule collation step. One of the more significant decisions within the rule process is the verification of whether an effected rule generates an inference result or not. Therefore, if a decision result is *'yes'*, the certainty values of affected evidence within the FEMS would then be updated accordingly.

Furthermore, it is necessary to determine whether an inference result has influence on the rule currently in effect, or whether the inference result affects another rule within the rule base. If the result of this decision is affirmative, the relevant rule(s) are adjusted accordingly and control within the flowchart is returned to the collated rule-set, that is, the start state.

The change of information flow to the start state when a rule is modified is an essential one. For instance, a rule may specify that all encrypted data files discovered on the storage media must be decrypted. However, such a rule may not be necessary (or even effected) if it occurs that no encrypted files are identified on the subject computer system. Alternatively, it may occur that the encryption strength applied on the identified files exceeds the capabilities of any augmented decryption solution employed within the FEMS.

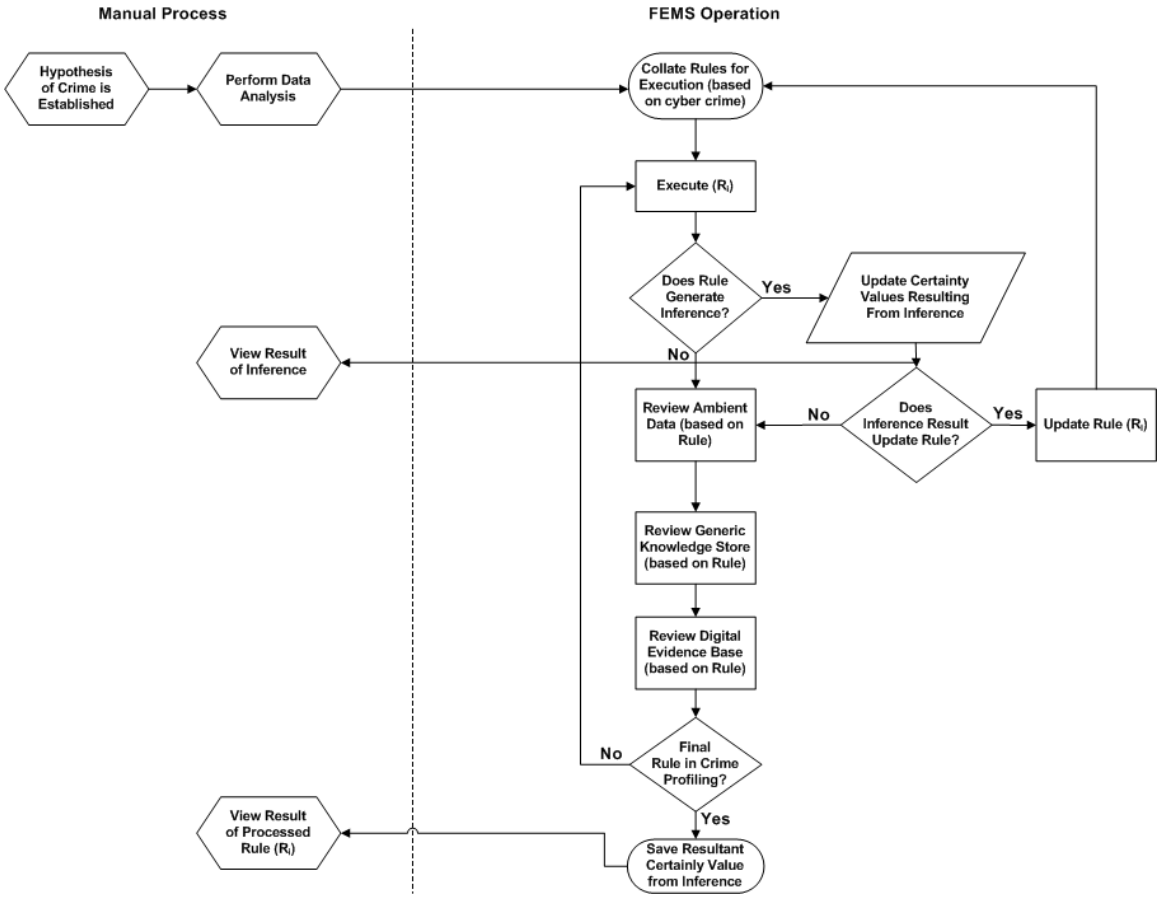

Figure 7.2: Flowchart for the Rule execution process

Subsequent to the two initial decisions within the rule process, the *Review* processes enable the application of a rule within the significant stores of the disk image. Thereafter, the consecutive rule within the rule-set is established and executed.

7.3.3 Decision state flowchart

The decision process flowchart commences when a rule requires an inference to be generated.

As depicted in Figure 7.3, the initial decision within this process is the confirmation of whether a rule is activated on or due to new evidence within the system. If the result of this decision is affirmative, control within the process then continues to

actions that update certainty values and rules that are affected by the introduction of the new evidence.

The initial decision phase is important because of the dynamic nature of forensic investigations. For example, during the data analysis phase, the introduction of a 'new' disk drive that is retrieved from the crime scene could have influence on investigative conclusions.

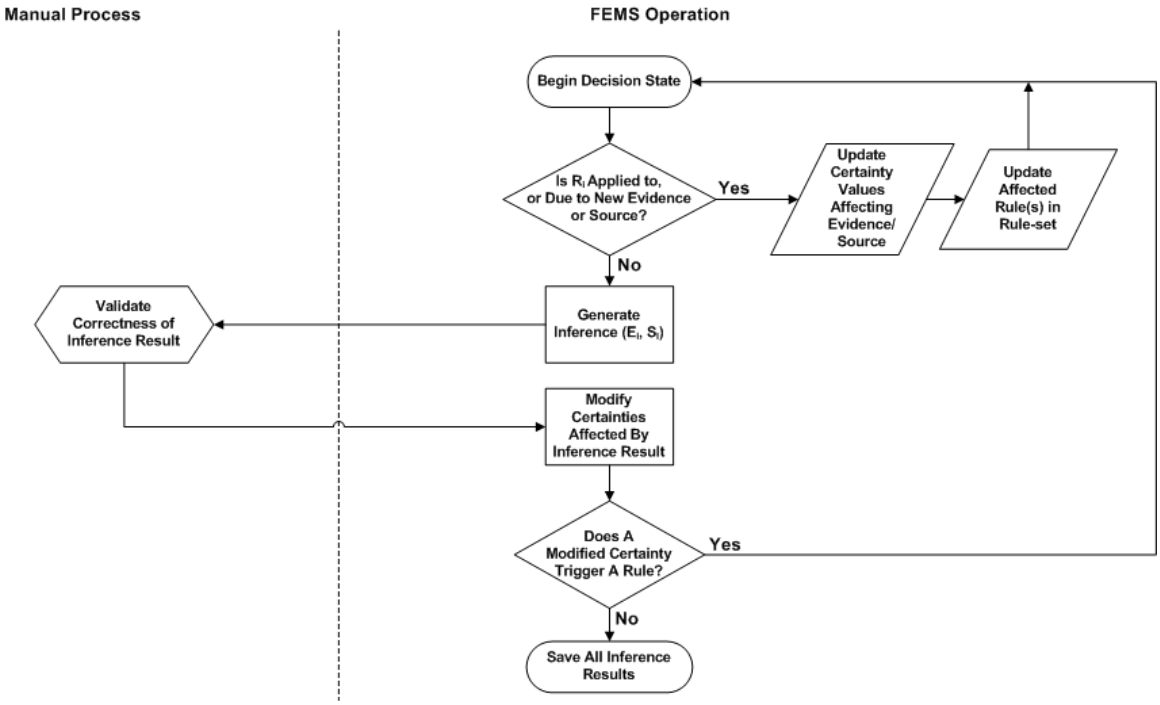

Figure 7.3: Flowchart for the Decision process

In the inference generation task, the system would typically perform a rudimentary analysis of evidence meta-data against that of pre-existing evidence. The outcome of this assessment would then be utilized to derive a conclusion. Similarly to other critical operations within the FEMS, the conclusion would then be offered to the investigator for further review and validation.

As depicted, specialist interpretation is only required once an inference is generated. This is necessitated by the system related tasks that follow such inferences, namely, the modification of certainties based on the inference and the decision on

whether a modified certainty activates a rule. Again, the importance of human expertise is emphasized in this task.

7.3.4 Data state flowchart

The data state exists to service all user and system-related queries to and within the FEMS respectively. We depict the data process flowchart in Figure 7.4.

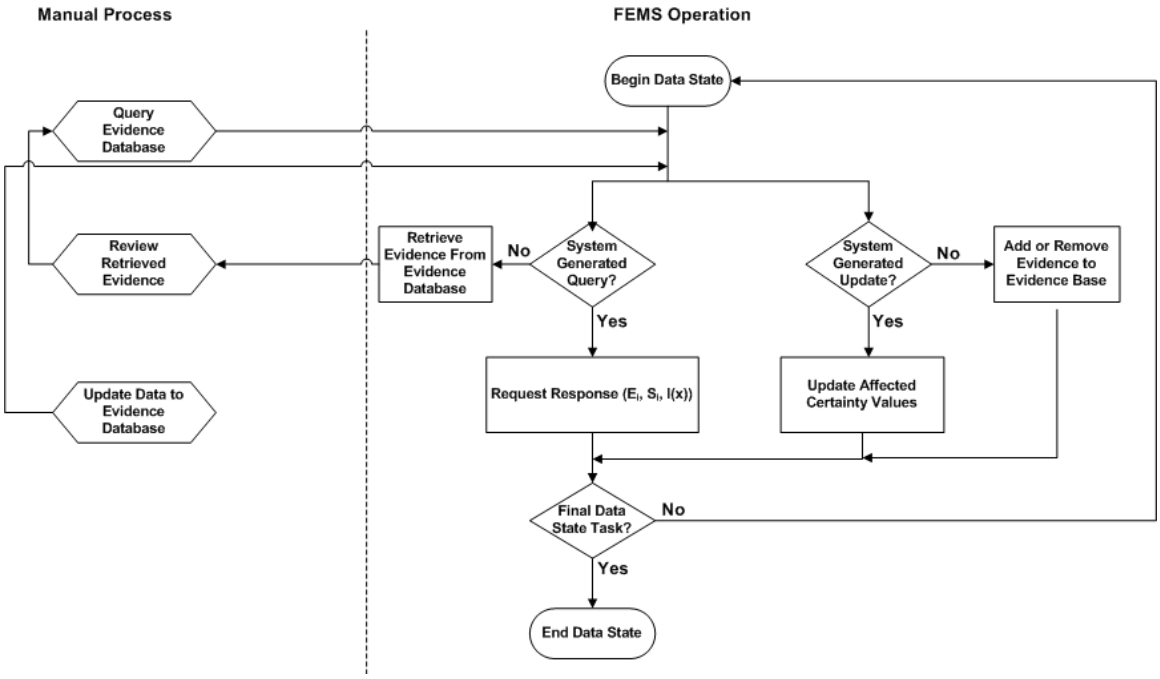

Figure 7.4: Flowchart for Data process

The data process flowchart is initiated when a system user (investigator) or system process queries or updates information within the FEMS. Subsequent to activation, the immediate consideration is whether such activation is user or programmatically generated. In the case of a user activated query, evidence would be retrieved from the evidence base and returned to the user for review. Where the query is system generated, the FEMS would return a request response (with the relevant data parameters) to the requesting system program. A similar rationale applies with evidence updates — where the update is user generated, evidence would be added to or removed from the evidence base. With a system generated update — which would

constitute a meta-data update — any certainty values associated with the update would be amended.

In this section we made use of flowcharts to elaborate on the information flow and component interactions within the FEMS's core components. This was done to provide a basis for deriving algorithms for these FEMS components. In the following section we provide the derived Hypothesis process, Rule process, Decision process and Data process algorithms.

7.4 FEMS component processing algorithms

The Hypothesis process, Rule process, Decision process and Data process algorithms are depicted below in Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4 respectively. Furthermore, we provide the reader with commentary regarding the algorithms below. As a result, we encourage a combined review of the relevant flowcharts provided in section 7.3 and the algorithms below in the course of the descriptions.

The algorithms are presented in pseudo-code; this approach was chosen to provide glimpses into programmatic details yet to be developed.

A step through the algorithms reveals the flow of transactions and information within the system. Although we utilize a C-like syntax, the logic within the pseudo-code, derived from the flowcharts provided earlier, can be represented in other programming languages or paradigms. Of particular interest is the control of transactions within Algorithm 2; the reader will notice that the variable i is reset to '0' within the code. The logic for this is that, if a rule (that is in effect) generates an inference, and that inference modifies any rules within the collated rule set, it is then necessary for all the rules within the rule set to be re-executed.

In Algorithm 3, we assume that the value of variable *execute_decision* is set programmatically when the decision process commences. In instances where new evidence or an inference result affects a rule within the Rule Base, the rule is then updated or invoked accordingly. For example, if a rule is configured to execute on evidence of C3 or lower, and an inference result now requires that the rule is executed on evidence of C4 and lower, then such a rule would be updated. Alternatively, if a

new log file is added to the Data Store, it may invoke a rule which is configured to assign certainty levels to that specific type of log file.

Similarly to Algorithm 3, in Algorithm 4, the value of variable *task* and *system_request* are set programmatically; the *task* variable is responsible for specifying whether the action to be performed is a *query* or an *update*, while the *system_request* variable specifies whether the task at hand is system generated or not. The result of *system_request* is also a significant control mechanism within the data process algorithm — if the value of this variable is set to *false*, then the task is interpreted as a user (investigator) generated task. Lastly, from a system perspective, we consider a polling mechanism to determine whether a request needs to be serviced by the Data Store. In instances where the task is user generated, the user is then required to confirm any further task(s).

7.5 Chapter summary

This chapter focused on the development of processing algorithms for the Hypothesis state, Rule state, Decision state and Data state of the Forensic Evidence Management System (FEMS), where flowcharts depicting the flow and control of information within the said state processes were developed. Two significant limitations are noticeable from the use of flowcharts: firstly, significant details cannot readily be portrayed within the diagrams. Secondly, the use of natural language poses a risk — natural language is often subjective, unlike mathematical notation, which is precise in its descriptions and hence interpretation. Nevertheless, such flowcharts and processing algorithms provide the foundation for a future implementation of the FEMS (or subsets of the system).

Although the purpose of this chapter is achieved, the algorithms provided could be better refined and more consideration could be placed on data structures, complexity and general analysis of these algorithms.

Algorithm 1 Hypothesis process pseudo-code

```
1 int task
2
3 print ("enter input character or EoF character")
4
5 while ((task = GetInput()) ≠ EoF character))
6 switch (task) /* application functionality selection phase */
7         case '1 : Review Information' /* information query functionality */
8                 print ("enter information request")
9                 Evidence ← GetInput()
10                Source ← GetInput()
11                Result ← REQUEST(Evidence_i, Source_i, I(x))
12                if (Result ≠ NULL)
13                then
14                        print result(s) to screen
15                        break;
16                else
17                        print ("no data to return")
18                        break;
19        case '2 : Rule' /* rule management functionality */
20                Display Rule amendment interface
21
22                /* determines whether a new rule modifies any certainty values */
23                if (certainty_value_updated(Rule_i))
24                then
25                        ∀ affected evidence in the Data Store
26                          Write new certainty value to Data Store
27                          break;
28                else
29                          Rule Base ← SAVE RULE(Rule_i)
30                          break;
31        case '3 : Modify Evidence' /* evidence management functionality */
32                Display fact amendment interface
33
34                /* determines whether new evidence modifies any certainty values */
35                if (certainty_value_updated(Evidence_i))
36                then
37                        ∀ affected evidence in the Data Store
38                          Write new certainty value to Data Store
39                          break;
40                else
41                          Data Store ← SAVE FACT(Evidence_i, Source_i, I(x))
42                          break;
```

Algorithm 2 Rule process pseudo-code

```
1 int i
2
3 /* function to generate an inference, given an FEMS object as input */
4 boolean generate_inference(System Obj)
5
6 i ← 0 /* counter indicating a rule that is in effect */
7 Collate rules for execution
8 while (i ≤ number of rules to be executed)
9       EXECUTE(Rule_i)
10
11      /* determines whether the current rule being processed
12      generates an inference */
13      if (generate_inference(Rule_i))
14      then
15            ∀ affected evidence and sources in the Data Store
16              Write new certainty value to Data Store
17
18            /* determines whether an inference generated by the current rule
19            modifies any other rules */
20            if ((generate_inference(Rule_i)) updates (Rule_x))
21            then
22                  Rule Base ← SAVE RULE(Rule_x)
23                  i ← 0 /* counter reset to re-execute all collated rules */
24            end if
25      else
26            Review ambient data, based on Rule_i
27            Review Generic Knowledge Store, based on Rule_i
28            Review Digital Evidence Base, based on Rule_i
29
30            /* increment to process the next rule within the collated rule-set */
31            i ← i + 1
32      end if
```

Algorithm 3 Decision process pseudo-code

```
1 boolean execute_decision
2
3 /* repetition condition for the decision module, set programmatically */
4 execute_decision ← execute_decision()
5
6 while (execute_decision)
7       if New_Evidence
8       then
9
10            /* set initial certainties for new evidence and sources */
11            ∀ new evidence
12              Write new certainty value of (E_i, S_i) to Data Store
13              ∀ rules in Rule Base
14
15                /* if new evidence changes or causes a rule to be invoked */
16                if (New Evidence (Evidence_i, Source_i) affects Rule_i)
17                then
18                      Rule Base ← SAVE RULE(Rule_i) or EXECUTE(Rule_i)
19                end if
20      else
21
22            /* function to generate an inference, given an FEMS
23            object as input */
24            generate_inference(System Obj)
25            ∀ evidence in Data Store, modify certainties affected by
26            inference result
27
28            /* determines whether an inference generated by the
29            current rule modifies any other rules */
30            if ((generate_inference(Rule_i)) affects (Rule_x))
31            then
32                  Rule Base ← SAVE RULE(Rule_i) or EXECUTE(Rule_i)
33            else
34                  Save inference result to Data Store
35                  execute_decision ← FALSE
36            end if
37      end if
```

Algorithm 4 Data process pseudo-code

```
1 char task /* specifies whether the task is a Query or an Update */
2 boolean system_request /* specifies whether the task is system generated */
3
4 /* task and system_request variables are initialized programmatically */
5 task ← task()
6 system_request ← system_generated()
7 while (task ≠ EoF character)
8         if (task is Query)
9         then
10                if (system_request)
11                then
12                        Return result to calling system process
13                        task ← set_new_system_task()
14                else
15
16                        /* a user (investigator) generated task */
17                        Retrieve evidence from evidence database
18                        Return result to fact retrieval interface
19                        print ("enter required task or EoF character")
20                        task ← task()
21        else
22
23                /* executes if task is an Update */
24                if (system_request)
25                then
26                        Update all affected certainty values
27                        task ← set_new_system_task()
28                else
29                        Add or Remove evidence from evidence base
30                        task ← task()
```

Chapter 8

A Comparison

8.1 Introduction

In his PhD thesis titled "*A Hypothesis-Based Approach To Digital Forensic Investigations*", Brian D Carrier [12] formally defines a digital forensic investigation and 31 unique classes of analysis techniques, which are ordered into 7 categories of digital forensic investigation analysis techniques. In his work, these definitions are based on an extended finite state machine (FSM) model, designed to include support for removable devices and complex states and events.

In this chapter we compare the constructs and operation of the Forensic Evidence Management System (FEMS) against the published work of Carrier [12].

The motivation for this approach is as follows: upon preliminary review of Carrier's work, parallels in problem definition and problem solution methodology between our works become apparent. For that reason, an assessment of the FEMS, against this published and accepted work is ideal in considering the 'completeness' of the proposed FEMS.

The aim of this chapter is therefore to provide a concise overview of the work presented by Carrier and to identify the core similarities and differences between our work. The result of this assessment is a validation of the solution approach in this dissertation and the constructs of the FEMS.

The remainder of this chapter is structured as follows: section 8.2 describes the

problem that is addressed within Carrier's work. Section 8.3 describes the solution utilized to address Carrier's problem statement and examples of the solution approach are provided. In section 8.4 we highlight the core similarities and differences between this work and Carrier's. However, wherever apt, such similarities and differences are highlighted in the course of the chapter. A chapter summary is then presented in section 8.5.

8.2 The problem

The motivation for Carrier's work hinges on the lack of formal theory relating to digital forensic investigation process. For instance, a practitioner in the field is able to describe how (s)he recognizes evidence, given a specific type of incident. However, this recognition process cannot usually be described in a general way, or in a formal and scientific language.

It is also noted that digital investigation process is currently steered by the technology being investigated and the available tools; although this focus is able to solve today's crimes, Carrier notes that the approach is limiting when the longer-term needs of the field are considered.

To further motivate the need for theory within investigation process, Carrier draws attention to the use of Daubert guidelines in various American states whenever scientific or technical evidence is submitted at a court of law. The four Daubert guidelines are as follows:

- Has the procedure been published (preferably in a journal)?
- Is the published procedure accepted by the relevant professional community?
- Can the procedure be tested?
- What is the error rate?

Based on these guidelines, it is apparent that, although the results of proprietary digital forensic solutions are often accepted within courts, such forensic tools may

not be as readily accepted if tested against these guidelines. In reality, the internal procedures of these forensic solutions are not publicly tested or formally published, albeit that there are valid reasons for this state of affairs.

The need for formalization is further supported by the condition that certain investigative practices — such as lead analysis in bullets — have been discontinued due to a lack of empirical evidence confirming the significance of such analysis results.

8.3 The solution approach

The first major contribution in Carrier's work is the design of a model to describe the concept of a computer's history — which contains the primitive and complex states and events that existed and occurred — thereby providing a general theory to test investigation hypotheses. The following requirements were used towards the model's design:

- The model must be based on the theoretical foundations of computing so that existing and future work in computer science can be used.
- The model must be general with respect to the technology being investigated so that the theory will apply to future as well as current technologies.
- The model must be capable of supporting events and storage locations at arbitrary levels of abstraction so that complex systems can be represented.
- The model must be capable of supporting systems with removable storage and event devices.
- The model must be capable of describing previous events and states so that all evidence can be represented.

The second major contribution in Carrier's work is to utilize the model in defining 31 unique classes of analysis techniques, which are then organized into seven categories. The following requirements were considered in developing the seven categories:

- The categories must be general with respect to the investigation technology so that new techniques can be identified and supported.
- The categories must be general with respect to the types of investigations and apply to law enforcement, industry, and military so that common terminology can be used.
- The categories must be specific so that general requirements can be defined to direct testing and development efforts.

In developing his model, Carrier makes extensive use of a classic computation theory model, namely a finite state machine (FSM). The main assumption in this regard is that the system (M) being investigated can be represented by a FSM quintuple $M = (Q, \Sigma, \delta, s0, F)$, where Q is a finite set of machine states and Σ is a finite alphabet of event symbols. The transition function $\delta : Q \times \Sigma \rightarrow Q$ is the event mapping between states in Q for each event symbol in Σ. The machine state changes only as a result of a new input symbol. The starting state of the machine is $s0 \in Q$ and the final states are $F \subseteq Q$.

Further to this, a *digital system* is defined as a connected set of digital storage and event devices; digital storage devices are physical components that can store one or more values, and a digital event device is a physical component that can change the state of a storage location. The *state* of a system is the discrete value of all storage locations, and an *event* is an occurrence that changes the state of the system. Therefore, the *history* of a digital system describes the sequence of states and events between two times.

To account for the changing system, functions that map a time to the value of a FSM variable are defined:

- $\Sigma(t)$ is the symbol alphabet of the FSM at time $t \in T$.
- $Q(t)$ is the set of all possible states of the FSM at time $t \in T$.
- $\delta(t)(s,e)$ is the transition function of the FSM at time $t \in T$ for state $s \in Q(t)$ and event $e \in \Sigma(t)$.

To this end, a graphical illustration of the above definitions is provided in Figure 8.1; the figure illustrates an event sequence for three time steps. The boxes with Rx are used to depict storage locations (registers in this instance). The circles represent an event that reads one or more registers and writes to one or more registers. Therefore, the history of this system includes the states and events that existed at each time step t.

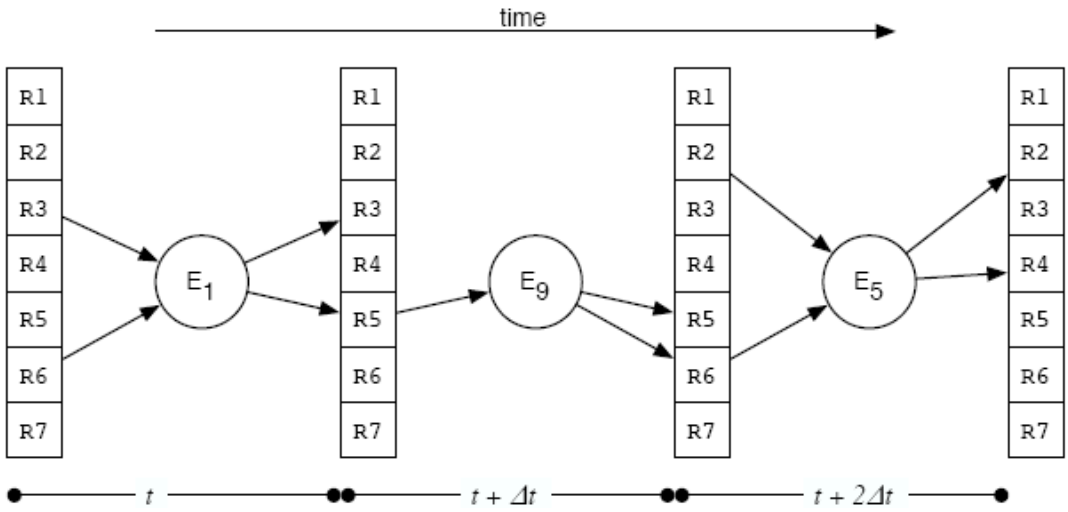

Figure 8.1: Representation of a sequence of events where the history of the system includes the events and state at each time [12]

We now provide preliminary information on the mathematical notation utilized in describing Carrier's history models.

In the following subsections, the names of mathematical sets will have all uppercase characters and the names of functions will have all lowercase characters. The first letter of the name is based on what the set or function is about. For example, the set "DAD" is about device "D" addresses "AD".

Sets and functions are defined for both the primitive and complex systems and the possible states and events; if the set or function is for the primitive system, then the first letter of the subscript in the set or function name is a "p". Where the set or function is for the complex system, then the first letter of the subscript in the set or function name is a "c". Furthermore, the second letter of the subscript is an "s"

if the set or function is for the system's state; the second letter of the subscript is an "e" if the set or function is for the system's events.

In describing the history models in the following subsections, we adopt the definitions of a *system*, a *state*, an *event* and the *history* of a system provided earlier in this section.

8.3.1 The Ideal Primitive History Model

The primitive history model for a system is formally defined by a tuple with eleven variables:

$$\mathbf{(T,\ D_{ps},\ DAD_{ps},\ ADO_{ps},\ c_{ps},\ h_{ps},\ D_{pe},\ DSY_{pe},\ DCG_{pe},\ c_{pe},\ h_{pe})}$$

T is the set of consecutive time values for which the history is defined. The D_{ps}, DAD_{ps}, ADO_{ps} and c_{ps} sets and functions describe the storage device capabilities and when they were connected to the system. The h_{ps} function is the primitive state history function. The D_{pe}, DSY_{pe}, DCG_{pe} and c_{pe} sets and functions describe the event devices and corresponding state changes and when they were connected to the system. The h_{pe} function is the primitive event history function.

In the system described by the tuple above, the set D_{ps} would contain the unique names of devices that were ever connected to the system. For instance, if a system under forensic investigation had only the CPU registers, memory, two hard disks and a removable USB device, then its D_{ps} set would contain the following:

$$\mathbf{\{CPU,\ memory,\ harddisk1,\ harddisk2,\ USB1\}}$$

The range of addresses for each device are defined in the DAD_{ps} set. That is, each entry in set DAD_{ps} is a set of addresses that are supported by a device. The set DAD_{ps} is therefore defined as:

$$\mathbf{DAD_{ps} = \{\{a : a \text{ is an address in device } d\} : d \in D_{ps}\}}$$

The function dad_{ps} maps a storage device name to its set of addresses. The dad_{ps} function is therefore defined as:

$$\mathbf{dad_{ps} : D_{ps} \rightarrow DAD_{ps}}$$

For example, dad_{ps}(harddisk1) could map to the values $\{0, 1, ..., 2^{31} - 1\}$ for a 2GB hard disk.

At this point, the descriptions provided of the primitive history model are sufficient for the purposes in section 8.4. We therefore refer the reader to [12] for a full description of the remaining variables in the eleven-tuple above.

A description of the ideal complex history model is provided in the following subsection.

8.3.2 The Ideal Complex History Model

Section 8.3.1 provided a model for the low-level events and storage locations for a computing system. However, it is understood that real world systems are complex and therefore provide several layers of data and event abstractions to hide the low-level details. As a result, a complex history model is required to accommodate such system abstractions. Ultimately, a digital forensic investigator tasked to analyse a computing system is typically concerned with complex storage locations, such as files.

The complex history model for a system is formally defined by a tuple with seventeen variables:

(T, L, D_{cs}, DAD_{cs}, DAT_{cs}, ADO_{cs}, ABS_{cs}, MAT_{cs}, c_{cs-X}, h_{cs}, D_{ce}, DSY_{ce-X}, DCG_{ce-X}, ABS_{ce}, MAT_{ce}, c_{ce}, h_{ce})

The set T is the same as described in section 8.3.1 and the set L contains the names of the complex abstract layers. An example of an L set is **{user, impl}**, where *user* represents a user layer and *impl* represents an implementation layer. The D_{cs}, DAD_{cs}, DAT_{cs}, ADO_{cs}, ABS_{cs}, MAT_{cs} and c_{cs-X} sets and functions are used to define the complex storage system; they also define the attribute and transformation functions for each complex storage location type. The D_{ce}, DSY_{ce-X}, DCG_{ce-X}, ABS_{ce}, MAT_{ce} and c_{ce} sets and functions are used to define the complex event system.

The set and function names for the complex history model are similar to those defined for the primitive history model (in section 8.3.1); the models only differ in the first letter "c" in the subscript. The D_{cs} set contains the unique names of the complex storage types that are supported by the system. It is noted that a system may contain data for a complex storage type. However, the system may not be able to process and transform it; if a computing system receives a file, but does not have

any programs to interpret it, then the complex storage type would not exist in the set D_{cs}, since the system has no way of using the data as a complex storage location.

The set DAT_{cs} contains an entry for every complex storage type within the system and the entry contains a list of attribute names. The set DAT_{cs} is defined as follows:

$$\mathbf{DAT}_{cs} = \mathbf{\{\{n : n \text{ is an attribute name for type } d\} : d \in D_{cs}\}}$$

In the complex history model, each complex storage type has one or more attributes. For example, the complex storage type may have attributes for *name*, *size* and *date*; the entry of such a complex storage type in DAT_{cs} would be {name, size, date}. Furthermore, the dat_{cs} maps a complex storage type to a set of attribute names. Such a mapping is defined as follows:

$$\mathbf{dat}_{cs} : \mathbf{D}_{cs} \rightarrow \mathbf{DAT}_{cs}$$

Once more, we refer the reader to [12] for a full description of the remaining variables in the seventeen-tuple above.

In the following section, we highlight the core similarities and differences between Carrier's work and the work presented in this dissertation.

8.4 The similarities and differences

In arriving at this stage, the description of Carrier's work in section 8.3 has been lengthy. However, this background description was a necessary precursor for the comparison exercise in this section.

The first, and possibly most noteworthy similarity between our work is the recognition of the dichotomy between investigations in the physical world, as apposed to the digital world. Carrier states the following in this regard: "*The goal of a digital investigation is to make valid inferences about a computer's history. Unlike the physical world, where an investigator can directly observe objects, the digital world involves many indirect observations. The investigator cannot directly observe the state of a hard disk sector or bytes in memory. He can only directly observe the state of output devices. Therefore, all statements about digital states and events are hypotheses that must be tested to some degree.*"

Carrier continues to define a *digital investigation* as a process that formulates

and tests hypotheses to answer questions about digital events or the state of digital data. Thereafter, he defines *digital evidence* as digital data that supports or refutes a hypothesis about digital events or the state of digital data. Therefore, an object is evidence if it contains information about events that occurred before, during, or after the incident being investigated.

At this juncture, it is apparent that the next similarity in our work is that we are cognisant of, and emphasise the role of hypotheses during the course of an investigation. From a FEMS perspective, this is especially significant in that, the system interface (within the client layer) allows an investigator to enter hypotheses — queries or facts — into the system during the course of an investigation. Furthermore, the finite state automaton (FSA) provided in Figure 6.1 depicts a hypothesis state within the system, which acts as the start and end state within the FSA.

There is no theoretical or technical difference between a Finite State Machine (FSM) and a Finite State Automata (FSA). As a result, a fundamental similarity between our work is in the use of these computational models; in his work, Carrier makes use of FSMs, while a FSA is utilized within this dissertation. Carrier utilizes finite state machines to define the concept of a computer's history (which contains the primitive and complex states and events that existed or occurred). In this dissertation we utilize a finite state automata to describe the component interactions and general operation of the forensic evidence management system. Nevertheless, in both contexts, these computational models incorporate the concept of system states, where previous system activity or "memory" is enshrined within such states. Ultimately, these finite state models are powerful tools for describing system history, since the goal of digital investigations is to make valid inferences about a computer's history.

An obvious difference between our work stems from the level of detail offered within our models — Carrier's work is detailed, contains mathematical rigour and is therefore robust. A review of sections 8.3.1 and 8.3.2 affirms this assertion. In fact, in sections of [12], theorems, as well as proofs of such theorems are provided. On the other hand, this work is abstract and emphasis is placed on providing descriptions of the system and the investigation logic, rather than the technicalities involved within the FEMS executing such logic. In the author's opinion, Carrier's contribution is an

effective display of the intricacies involved in formally describing a digital forensic investigation, coupled with the construction of feasible scenarios to test or reason about such formal descriptions.

In this dissertation, the Biba Integrity Model and Casey's Certainty Scale are explicitly incorporated into the construction of the FEMS; this is a subtle difference between our work.

Incorporating the Biba model and Casey's scale is fundamentally different to Carrier's work as a result of differing solution approaches. In this work, we introduce the Biba model as a mechanism for managing the integrity of digital evidence within the FEMS, where the integrity of evidence within the FEMS is protected using the two fundamental properties of the Biba model. In contrast, Carrier's approach presents no requirement for the preservation of evidential integrity; this is understandable, in view of the fact that the models in Carrier's work focus on formally defining a mechanism proving the existence of historical system events.

Our use of Casey's certainty scale introduces the concept of trustworthiness of digital evidence and the sources thereof. Conversely, the concept of certainty is not incorporated within Carrier's work. However, in a section on future work, Carrier identifies the matter of certainty values as an area for further development; Carrier acknowledges the need for assigning certainty within event reconstruction — that is, investigations — and concedes to the inherent complexity associated with deriving such certainty values. He states the following in this regard, "*It is not clear how to assign certainty values to these hypotheses or how to measure the notion of trust so that independent parties can compare the amount of trust they have in a component*". In the same text, Carrier then highlights the potential usefulness of Casey's certainty scale, as well as the shortcomings of Casey's approach.

Lastly, a core similarity between our work relates to the significance of log evidence towards investigations. In this dissertation, we affirm this significance within Chapter 4, which is dedicated to the topic of logging and log correlation. Carrier states the following in this regard, "*When event and state reconstruction are considered, it becomes clear that without reliable logs that record which events occurred then low certainty values exist because there is typically not a unique path to each state*".

This view highlights the significance of certainty of investigative conclusions and the need for reliable log evidence.

8.5 Chapter summary

The aim of this chapter was to compare the work in this dissertation against previously published (and comparable) work. In particular, work in this dissertation is compared to work presented by Carrier in his PhD thesis titled "*A Hypothesis-Based Approach To Digital Forensic Investigations*". In so doing, we were able to scrutinize the FEMS model, thereby revealing the notable similarities and differences between Carrier's model and the model presented in this dissertation.

A fundamental similarity between our work was in the use of computational models to describe the operation of our systems — Carrier made use of FSMs to define a theoretical model for a computer's history. The FSMs were then used to test investigative hypotheses against the states and event variables as described within the computer's history model. On the other hand, we proposed and described the constituents of an integrity-aware forensic evidence management system. Thereafter, we made use of a FSA to illustrate the flow of information and the core component interactions within the FEMS system (during a forensic investigation).

A notable difference within the solution presented in this dissertation was in the consideration of an integrity classification model — the Biba Integrity model — and the use of Casey's certainly values as the integrity classification scheme. Conversely, Carrier's work contained thorough mathematical descriptions of his models; this is an aspect that was deficient in this work.

The value in the comparative approach in this chapter lies in the fact that the examination exercise has highlighted positive elements in the FEMS model and revealed some shortcomings as well. Some of the shortcomings within our work can be deemed as areas for future research. However, it is interesting to note that some of the shortcomings in this work — such as the calculation and assignment of certainty values — were also highlighted in Carrier's work as areas for further development

Chapter 9

Conclusion

9.1 Summary

In the current information age, commercial, medical and even academic institutions make use of information as a competitive advantage and source for financial gain. Simultaneously, the protection of such information from 'the insider' threat or malicious external entities is a key consideration within such institutions. As a result, institutions continue to make use of administrative and technical controls to circumvent the ever-increasing threat of cyber-crime.

One of the notable interventions by concerned organizations has been the creation of, or increase in capacity within incident response and computer forensics capabilities; organizations recognize that, in addition to any detective or preventative controls, it is equally important to recover from and identify the source of cyber-crimes. This requirement is however not without its challenges.

Some of the challenges resultant from technological advancement were highlighted in Chapter 1. For instance, the prevalence of mobile communications devices and the ease with which cyber-crimes could be perpetrated in the advent of the Internet were discussed. The challenges experienced within computer forensic investigations were also highlighted. For example, the lack of a consolidated view of an investigation landscape. Notably, the volatility of forensic evidence and the significance of evidential integrity were pronounced in this regard; this challenge formed the core of the

problem to be addressed by the construction of a integrity-aware Forensic Evidence Management System (FEMS).

On the whole, the FEMS is intended as a decision-support tool for the computer forensic investigator.

In Chapter 2 a distinction was drawn between *true* cyber-crime and *e-enabled* cyber crime, where Denial of Service attacks and credit card misuse are examples of such crimes respectively. Insight into the psychology and motivation of cyber criminals was also provided.

Chapter 3 laid the foundation upon which the proposed system was based — a cross-section of the art and science of computer forensics was provided. Milestones from the history of forensics, the creation of computer forensics, the fundamental principles and theories surrounding the field, the stages within the computer forensics investigation processes, as well as the instrumentation (specifically software tools) related to the field were provided.

Chapter 4 provided a glimpse into one of the key artifacts towards the operation of the FEMS — a system administrator and a computer forensic specialist's greatest ally — a log file. The message in this chapter was used to remind the reader of the significance of logging, the availability of log files in the event of an incident and the usefulness of log correlation. In addition, some of the log correlation techniques provided in this chapter — particularly rule-based correlation — were adapted and utilized in the construction of the FEMS.

As indicated earlier, Chapter 1 provided a wide view on the challenges within the field of computer forensics. However, the scope of this dissertation was limited to address the challenge related to evidential integrity within computer forensic investigations. In order to address this problem, we proposed the construction of an integrity-aware Forensic Evidence Management System (FEMS), with the following system requirements:

- to manage digital evidence and the integrity,
- to preserve the integrity of evidence within the system, and

- to provide a degree of automation within the analysis stage within forensic investigations.

The core contribution of this dissertation — the construction of the FEMS — was detailed throughout Chapter 5. In this chapter the FEMS architecture was illustrated and the components within the client, logic and data layers of the proposed system were described. Two principal inclusions within the system architecture was that of the Biba Integrity Model and Casey's Certainty Scale. The Biba model was incorporated to preserve the integrity of all evidentiary artifacts hosted within the system. A consequence from the use of the Biba model is the need for an integrity classification scheme. Due to its application within network environments, Casey's Certainty Scale was chosen and utilized as the integrity classification scheme within the FEMS.

Chapter 6 extended on the preliminary discussions and information flow descriptions on the FEMS provided in Chapter 5. In particular, we made use of Finite State Automata (FSA) theory to describe the core states of the FEMS, thereby describing the general behaviour and component interactions within the system. Furthermore, a core contribution within this chapter was the encapsulation of cyber-crime profiles within the Rule state of the FEMS FSA. With the use of a child exploitation and computer intrusion scenario, the states and transitions of the FEMS FSA were expounded further.

Chapter 7 delved into the mechanics of the states within the FEMS FSA, where flowcharts were utilized to describe the inner-workings of the FEMS FSA states. This was also done in an effort to provide input towards technicalities associated with the implementation of such a system. Furthermore, FEMS component processing algorithms were extrapolated from the flowcharts provided; this also provided some lower-level insight towards certain implementation considerations.

Lastly, in Chapter 8 we compared the FEMS with previously published work by Carrier. Carrier's work focused on the use of a hypothesis-based approach within digital investigations; he made use of finite state machines to model the history of a computer system under forensic investigation. Key definitions within Carrier's work relates to the definition of *history* and *digital investigations*; Carrier states that the

history of a system includes the states and events that existed at each time step t. In addition, Carrier defines a digital investigation as a process that formulates and tests hypotheses to answer questions about digital events or the state of digital data.

Therefore, the rationale for performing the comparison within this chapter was to identify conceptual, as well as 'implementation' similarities and differences between our work. As a result of the comparison, shortcomings within our solution approach, such as the limited mathematical rigour, as well as meaningful considerations, such as the incorporation of Casey's Certainly Scale were identified. The work presented by Carrier was indeed intriguing, and as far as the author could establish, was the most appropriate work to compare the FEMS against.

9.2 Limitations and future work

In this section we briefly discuss the limitations within the Forensic Evidence Management System (FEMS) architecture. In so doing, we point the reader towards areas for future research within the context of this work and the field of digital forensic investigations at large.

The application of the FEMS is largely proposed within a managed network environment. This approach was taken to minimize the complexity in discussions resulting from the application of such a system within a wider network (Internet) environment. However, such an assumption may not be practical when considering the source of a significant number of cyber-crimes. Furthermore, the usefulness of such a solution may extend well within certain aspects of larger network environments. This is therefore an ideal area for further exploration.

Although the FEMS and its components have been described within this work, the prototype implementation and proof of concept evaluation of such a system may not be simple (or even practical) — the multiplicity of input sources and the disparity in formats of such input sources could be a hindrance to the development and effective operation of the system. However, the prototype implementation of specific components of such a system may be feasible; perhaps such developments could be used as stand-alone utilities within the computer forensic investigator's toolkit. That being

said, other design aspects may only be revealed within an implementation exercise. Such an implementation exercise would necessitate the use of software engineering practices to arrive at formal system requirements, functional specifications, an implementation architecture and the boundary conditions under which the system is specified. This should therefore be considered in future work.

Integration between the FEMS and other relevant IT systems within network environments is required; a description of the technicalities involved in achieving the desired level of integration is certainly an area for further exploration.

Despite the provision of explanations and examples of the FEMS's operation, mathematical formulation and defence of its core concepts would have been ideal. Casey's Certainty Scale (CCS) is an ideal example of this point; further development in terms of its formality and testing is required. Such future research would certainly improve the soundness of arguments based on CCS.

Lastly, a facet beyond the scope of this work — but deserving of further study — was an analysis of error rates and proofs of correctness of the FEMS algorithms and the system as a whole. The usefulness of such a system could be diminished if the system error rate was high, or perceived to be high; ultimately, an investigator must trust the FEMS components (and the system at large) in order to trust the outputs of the system. Furthermore, the model does not extensively consider exception handling and how such exceptions will be catered for within the system.

Bibliography

[1] Emad Aboelela and Christos Douligeris. Fuzzy Temporal Reasoning Model for Event Correlation in Network Management. In *Conference on Local Computer Networks*, pages 150–159, October 1999.

[2] AccessData. Forensic Toolkit (FTK), Last accessed 6 July 2010. http://www.accessdata.com/forensictoolkit.html.

[3] Atif Ahmad and Anthonie B. Ruighaver. Improved Event Logging for Security and Forensics: Developing Audit Management Infrastructure Requirements. In *Proceedings of ISOneWorld*, April 2003.

[4] Kweku K. Arthur and Hein S. Venter. An Investigation Into Computer Forensic Tools. In *Proceedings of the Fourth Annual Information Security South Africa Conference (ISSA2004)*, June/July 2004. Published electronically.

[5] John Austen. Some stepping stones in computer forensics. *Information Security Technical Report*, 8(2):pp. 37–41, September 2003.

[6] British Broadcasting Corporation (BBC). Henry Faulds (1843-1930), Last accessed 7 July 2010. http://www.bbc.co.uk/history/historic_figures/faulds_henry.shtml.

[7] Chris Boyd and Pete Forster. Time and date issues in forensic computing—a case study. *Digital Investigation*, 1(1):pp. 18–23, February 2004.

[8] Kit Burden and Creole Palmer. Cyber Crime: A new breed of criminal? *Computer Law & Security Report*, 19(3):pp. 222–227, September 2003.

[9] Paul Burke and Philip Craiger. Forensic Analysis of Xbox Consoles. In P. Craiger and S. Shenoi, editors, *Advances in Digital Forensics III*, volume 242, pages 269–280. Boston: Springer, 2007.

[10] Luanne M. Burns, Joe L. Hellerstein, Sheng Ma, Chang-Shing Perng, David A. Rabenhorst, and David J. Taylor. Towards Discovery of Event Correlation Rules. In *IEEE/IFIP International Symposium on Integrated Network Management Proceedings*, pages 345–359, May 2001.

[11] Fethi Calisir and Cigdem A. Gumussoy. Internet banking versus other banking channels: Young consumers' view. *International Journal of Information Management*, 28(3):pp. 215–221, June 2008.

[12] Brian D. Carrier. *A Hypothesis-Based Approach To Digital Forensic Investigations*. PhD thesis, Purdue University, May 2006. Pages 190.

[13] Eoghan Casey. Error, Uncertainty, and Loss in Digital Evidence. *International Journal of Digital Evidence*, 1(2), Summer 2002.

[14] Mohamed Chawki. The Digital Evidence in the Information Era, March 2004. http://www.crime-research.org/articles/chawki1/, Last accessed 6 July 2010.

[15] Wingyan Chung, Hsinchun Chen, Weiping Chang, and Shihchieh Chou. Fighting cybercrime: a review and the Taiwan experience. *Decision Support Systems*, 41(3):pp. 669–682, March 2006.

[16] Anton Chuvakin. Event Correlation in Security, Last accessed 6 July 2010. http://www.securitydocs.com/library/2270.

[17] Séamus Ó. Ciardhuáin. An Extended Model of Cybercrime Investigations. *International Journal of Digital Evidence*, 3(1), Summer 2004.

[18] Cable News Network (CNN). A convicted hacker debunks some myths, October 2005. http://www.cnn.com/2005/TECH/internet/10/07/kevin.mitnick.cnna/, Last accessed 6 July 2010.

[19] Fred Cohen. Computer Viruses: Theory and Experiments. *Computers & Security*, 6(1):pp. 22–35, February 1987.

[20] Fred Cohen. On the Implications of Computer Viruses and Methods of Defense. *Computers & Security*, 7(2):pp. 167–184, April 1988.

[21] crimeZZZ.net. History of forensic science, Last accessed 7 July 2010. http://www.crimezzz.net/forensic_history/.UNUSED/text.htm.

[22] Adrian Culley. Computer forensics: past, present, and future. *Information Security Technical Report*, 8(2):pp. 32–36, 2003.

[23] Neil Docherty. interview: anonymous, Last accessed 6 July 2010. http://www.pbs.org/wgbh/pages/frontline/shows/hackers/interviews/anon.html.

[24] Editorial. Napster back in court. *Network Security*, 2001(10):pp. 2, October 2001.

[25] David Emm. Focus on trojansholding data to ransom. *Network Security*, 2006(6):pp. 4–7, June 2006.

[26] Dario V. Forte. The "Art" of log correlation: Tools and Techniques for Correlating Events and Log Files. *Computer Fraud & Security*, 2004(8):pp. 15–17, August 2004.

[27] Robert H. Frank. *Microeconomics and Behaviour*. International Editions. McGraw-Hill, Third edition, 1998. pp. 342.

[28] Joseph Giordano and Chester Maciag. Cyber Forensics: A Military Operations Perspective. *International Journal of Digital Evidence*, 1(2), Summer 2002.

[29] Sarah Gordon and Richard Ford. On the definition and classification of cybercrime. *Journal in Computing Virology*, 2(1):pp. 13–20, August 2006.

[30] Jivesh Govil and Jivika Govil. Ramifications of Cyber Crime and Suggestive Preventive Measures. In *IEEE International Conference on Electro/Information Technology*, pages 610–615, 2007.

[31] David Groth and Dan Newland. *A+ Complete Study Guide*. Sybex Inc., Second edition, 2001.

[32] Shon Harris. *All in One CISSP Certification Exam Guide*. McGrawHill-Osborne Press, 2002.

[33] Linda Harrison. Bedroom NASA hacker set to bite pillow in choky, September 2000. http://www.theregister.co.uk/2000/09/22/bedroom_nasa_hacker_set/, Last accessed 6 July 2010.

[34] Simon Heron. Gang culture in the online world. *Network Security*, 2007(11):pp. 4–7, November 2007.

[35] Stephen Hinde. Cyber-terrorism in context. *Computers & Security*, 22(3):pp. 188–192, April 2003.

[36] Chet Hosmer. Proving the Integrity of Digital Evidence with Time. *International Journal of Digital Evidence*, 1(1), Spring 2002.

[37] HTCIA. High Technology Crime Investigation Association, Last accessed 6 July 2010. http://www.htcia.org/.

[38] Vafa D. Izadinia. Fingerprinting Encrypted Tunnel Endpoints. Master's thesis, University of Pretoria, February 2005. Pages 146.

[39] Andrew Jones. Industrial espionage in a hi-tech world. *Computer Fraud & Security*, 2008(1):pp. 7–13, January 2008.

[40] Paul E. Jones. US Secure Hash Algorithm 1 (SHA1), September 2001. http://www.ietf.org/rfc/rfc3174.txt, Last accessed 6 July 2010.

[41] Panagiotis Kanellis, Evangelos Kiountouzis, Nicholas Kolokotronis, and Drakoulis Martakos, editors. *Digital Crime and Forensic Science in Cyberspace*. Idea Group Publishing, 2006.

[42] Erin E. Kenneally. Digital logs—proof matters. *Digital Investigation*, 1(2):pp. 94–101, June 2004.

[43] Sherif Khattab, Rami Melhem, Daniel Mossé, and Taieb Znati. Honeypot back-propagation for mitigating spoofing distributed Denial-of-Service attacks. *Journal of Parallel and Distributed Computing*, 66(9):pp. 1152–1164, September 2006.

[44] Maria Kjaerland. A taxonomy and comparison of computer security incidents from the commercial and government sectors. *Computers & Security*, 25(7):pp. 522–538, October 2006.

[45] Jaco Kroon and Martin S. Olivier. An Approach to Building a Fortified Network Logger That is Resilient to Cracking. In *Proceedings of the Fourth Annual Information Security South Africa Conference (ISSA2004)*, June/July 2004. Published electronically.

[46] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, 2001.

[47] Kang-Won Lee, Suresh Chari, Anees Shaikh, Sambit Sahu, and Pau-Chen Cheng. Improving the resilience of content distribution networks to large scale distributed denial of service attacks. *Computer Networks*, 51(10):pp. 2753–2770, July 2007.

[48] Ryan Leigland and Axel W. Krings. A Formalization of Digital Forensics. *International Journal of Digital Evidence*, 3(2), Fall 2004.

[49] Steven D. Levitt and Stephen J. Dubner. *Freakonomics*. Penguin Books, 2005. pp. 59-62.

[50] Andrew M. Lister and Robert D. Eager. *Fundamentals of Operating Systems*. Macmillan Press, Fifth edition, 1993.

[51] James R. Lyle. NIST CFTT: Testing Disk Imaging Tools. *International Journal of Digital Evidence*, 1(4), Winter 2003.

[52] John C. Martin. *Introduction to Languages and the Theory of Computation*. Programming Languages Series. McGraw-Hill International Editions, Second edition, 1997.

[53] Rodney McKemmish. What is forensic computing? In *Australian Institute of Criminology. Trends and issues in crime and criminal justice*, pages 37–47, June 1999.

[54] Brian McKenna. Russian zombie army turned back. *Computer Fraud & Security,*, 2004(7):pp. 1–2, July 2004.

[55] Matthew Meyers and Marc Rogers. Computer Forensics: The Need for Standardization and Certification. *International Journal of Digital Evidence*, 3(2), Fall 2004.

[56] Bimal K. Mishra and Dinesh Saini. Mathematical models on computer viruses. *Applied Mathematics and Computation*, 187(2):pp. 929–936, April 2007.

[57] Soumyo D. Moitraa and Suresh L. Konda. An empirical investigation of network attacks on computer systems. *Computers & Security*, 23(1):pp. 43–51, February 2004.

[58] Tyler Moore. Phishing and the economics of e-crime. *Infosecurity*, 4(6):pp. 34–37, September 2007.

[59] Rod Morris. Options in Computer Forensic Tools. *Computer Fraud & Security*, 2002(11):pp. 8–11, November 2002.

[60] Andre Muscat. A Log Analysis based Intrusion Detection System for the creation of a Specification based Intrusion Prevention System. In *Proceedings of the 2003 University of Malta Computer Science Annual Research Workshop*, July 2003.

[61] Francesco Nerieri, Radu Prodan, and Thomas Fahringer. Kalipy: A Tool for Online Performance Analysis of Grid Workflows through Event Correlation. In *The Second IEEE International Conference on e-Science and Grid Computing*, page 18, December 2006.

[62] National Institute of Science and Technology. National Software Reference Library, Last accessed 6 July 2010. http://www.nsrl.nist.gov/.

[63] Yongseok Park. Event Correlation. *IEEE Potentials*, 20(2):pp. 34–35, April/May 2001.

[64] Ahmed Patel and Seamus Ciardhuain. The Impact of Forensic Computing on Telecommunications. *IEEE Communications Magazine*, 38(11):pp. 64–67, November 2000.

[65] Technology Pathways. ProDiscover, Last accessed 6 July 2010. http://www.techpathways.com/DesktopDefault.aspx?tabindex=0&tabid=1.

[66] Charles P. Pfleeger and Shari L. Pfleeger. *Security in Computing*. Professional Technical Reference. Prentice Hall, Third edition, 2003.

[67] J Postel. Internet Protocol: DARPA Internet Program Protocol Specification, September 1981. http://www.rfc-editor.org/rfc/rfc791.txt, Last accessed 6 July 2010.

[68] J Postel. Transmission Control Protocol, September 1981. http://www.rfc-editor.org/rfc/rfc793.txt, Last accessed 6 July 2010.

[69] Richard Power and Dario Forte. Information age espionage, debriefing a real-world, top-class cyber sleuth. *Computer Fraud & Security*, 2007(5):pp. 10–14, May 2007.

[70] Maria M. Pozzo and Terence E. Gray. An Approach to Containing Computer Viruses. *Computers & Security*, 6(4):pp. 321–331, August 1987.

[71] Robert Richardson. CSI Computer Crime and Security Survey, Last accessed 6 July 2010. http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf.

[72] Ronald L. Rivest. The MD5 Message-Digest Algorithm, April 1992. http://www.ietf.org/rfc/rfc1321.txt, Last accessed 6 July 2010.

[73] Marc Rogers. The role of criminal profiling in the computer forensics process. *Computers & Security*, 22(4):pp. 292–298, May 2003.

[74] Marcus K. Rogers and Kate Seigfried. The future of computer forensics: a needs analysis survey. *Computers & Security*, 23(1):pp. 12–16, February 2004.

[75] Robert Rowlingson. A Ten Step Process for Forensic Readiness. *International Journal of Digital Evidence*, 2(3), Winter 2004.

[76] Bruce Schneier and John Kelsey. Cryptographic Support for Secure Logs on Untrusted Machines. In *The Seventh USENIX Security Symposium Proceedings*, pages 53–62. USENIX Press, 1998.

[77] Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security*, 2(2):pp. 159–176, May 1999.

[78] Eugene E. Schultz. Internet security: what's in the future? *Computers & Security*, 22(2):pp. 78–79, February 2003.

[79] Billy Smith. Thinking About Security Monitoring and Event Correlation, Last accessed 7 July 2010. http://www.lurhq.com/research/articles/correlation.

[80] Guidance Software. EnCase, Last accessed 6 July 2010. http://www.guidancesoftware.com/.

[81] Ricardo C. Sousa and Lucian I. Prejbeanu. Non-volatile magnetic random access memories (MRAM). *Comptes Rendus Physique*, 6(9):pp. 1013–1021, November 2005.

[82] William Stallings. *Operating Systems: Internals and Design Principles*. Prentice Hall International, Fourth edition, 2001.

[83] Peter Stephenson. The right tools for the job. *Digital Investigation*, 1(1):pp. 24–27, February 2004.

[84] John Sterlicchi. Clinton wants $2 billion to fight cyber-terrorism. *Computer Fraud & Security*, 2000(2):pp. 6, February 2000.

[85] Cliff Stoll. How Secure are Computers in the U.S.A.?: An analysis of a series of attacks on Milnet computers. *Computers & Security*, 7(6):pp. 543–547, December 1988.

[86] Mark Sunner. The rise of targeted trojans. *Network Security*, 2007(12):pp. 4–7, December 2007.

[87] Carol Taylor, Barbara Endicott-Popovsky, and Deborah A. Frincke. Specifying digital forensics: A forensics policy approach. *Digital Investigation*, 4(1):pp. 101–104, September 2007.

[88] New Technologies. Cluster Defined, Last accessed 6 July 2010. http://www.forensics-intl.com/def19.html.

[89] New Technologies. Computer Forensics Defined, Last accessed 6 July 2010. http://www.forensics-intl.com/def4.html.

[90] Jan K. Tudor. *Information Security Architecture: An Integrated Approach to Security in the Organization*. Auerbach, First edition, 2001.

[91] United States National Institute of Justice. Electronic Crime Scene Investigation: A Guide for First Responders, July 2001. http://www.ncjrs.gov/pdffiles1/nij/187736.pdf, Last accessed 7 July 2010.

[92] David A. Volgas, James P. Stannard, and Jorge E. Alonso. Ballistics: a primer for the surgeon. *Injury*, 36(3):pp. 373–379, March 2005.

[93] Yun Wang, James Cannady, and James Rosenbluth. Foundations of computer forensics: A technology for the fight against computer crime. *Computer Law & Security Report*, 21(2):pp. 119–127, 2005.

[94] Carrie M. Whitcomb. An Historical Perspective of Digital Evidence: A Forensic Scientists View. *International Journal of Digital Evidence*, 1(1), Spring 2002.

[95] Wikipedia. Finite State Machine, Last accessed 6 July 2010. http://en.wikipedia.org/wiki/Finite_state_machine.

[96] Wikipedia. Gary McKinnon, Last accessed 6 July 2010. http://en.wikipedia.org/wiki/Gary_McKinnon.

[97] Jack Wiles. Computer forensics – yesterday, today and tomorrow, Last accessed 22 August 2008. http://www.infosecurity-magazine.com/comment/071003_wiles.htm.

[98] Henry B. Wolfe. Computer forensics. *Computers & Security*, 22(1):pp. 26–28, January 2003.

[99] Henry B. Wolfe. Evidence analysis. *Computers & Security*, 22(4):pp. 289–291, May 2003.