ABSTRACT

SREMACK, JOSEPH C. Formalizing Computer Forensic Analysis: A Proof-Based Methodology. (Under the direction of Dr. Mladen A. Vouk and Dr. Jun Xu).

Computer forensics is an important subject in the field of computer security. Impenetrably secure systems are not a reality - hundreds of thousands of security breaches are reported annually. When a security breach does occur, certain steps must be taken to understand what happened and how to recover from the incident, including data collection, analysis, and recovery. These responses to an incident comprise one part of computer forensics. A successful forensic investigation of any security breach requires a sound approach. Forensics literature provides a general model for conducting an investigation that can acts as a template for forensic investigations. The current literature, however, has primarily focused on two extremes of forensics: technical details and high-level procedural guidelines. By focusing on the extremes, many of the intermediate steps and logical conclusions that a forensic investigator must make are omitted. This omission leaves the burden of forming the logical structure of an investigation to the investigator. Such *ad hoc* approaches can lead to inefficient investigations with extraneous investigatory steps, and possibly less accurate results.

This thesis explores the formalization of existing computer forensic analysis techniques such that a complete forensic investigation can be conducted in an efficient and meticulous manner. The formalization includes the use of high-level incident information to formulate a broad hypothesis about the entire incident. The hypothesis is then proven by performing a series of lower-level proofs - either by inductive or by deductive (axiomatic inductive) means - each of which acts as a premise for the overall incident hypothesis. The formalized analysis is then applied to actual forensic investigations to demonstrate its effectiveness. The formalized methodology and techniques presented in this thesis demonstrate how forensic investigations can be scientifically rigorous without sacrificing the necessary amount of creativity that is required for a complete investigation.

Formalizing Computer Forensic Analysis: A Proof-Based Methodology

by

Joseph C. Sremack

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial satisfaction of the
requirements for the Degree of
Master of Science

Department of Computer Science

Raleigh

2004

Approved By:

Dr. Peng Ning
Committee Member

Dr. Mladen A. Vouk
Co-Chair of Advisory Committee

Dr. Jun Xu
Co-Chair of Advisory Committee

Biography

Joe Sremack originates from Canal Fulton, Ohio. He spent the first twenty-two years of his life living around Akron. His last four years in Ohio were spent at The College of Wooster, where he double-majored in computer science and philosophy. His computer science focuses were system programming and software design, and his philosophy focuses were Wittgenstein, logic, and the philosophy of language. These interests were combined for his senior thesis, in which he explored program verification and the inherent philosophical issues.

At N.C. State, Joe has primarily focused on computer security issues. He has spent the past two years studying computer forensics and software-level vulnerabilities. He also presented a paper on file system computer forensics entitled "Cross-Validation of File System Layers for Computer Forensics," at the Digital Forensics Research Workshop in August 2003.

In his spare time, Joe likes to play chess (especially "one-minute bullet"), exercise, and spend time with his girlfriend, Alicia.

Acknowledgements

This thesis would not be possible without the unwavering support and encouragement from several very important people. First, I must thank my parents for convincing me to "stick with it" when I experienced hardships and wanted nothing more than to throw in the towel. Once again, they are right, and I am glad they are. Thanks go out to my sister, Becky, for her editorial help. While I am sure that I could have heeded her advice more, this thesis is much better because of her help. I must also give my heartfelt gratitude to Dr. Mladen Vouk for all of the support and the many opportunities he gave me. Most other professors would not have given me even a fraction of the help that he gave to me.

I would also like to acknowledge the people of the Digital Forensics Research Workshop. Their work is paving the way for computer forensics to truly become a science, and it encouraged me to pursue an "abstract" thesis rather than tackling an immediate, technical problem. I would like to think that this thesis will find a place in their research.

This work has been supported in part by a grant from Cisco Systems and IBM's Shared University Research program.

Contents

List of Figures

Chapter 1

Introduction

Computer attacks cannot be fully prevented. No matter how many proactive security measures are in place - such as intrusion detection systems and firewalls - a skilled attacker can eventually perform a successful attack. This is apparent based on the number of incidents reported to CERT/CC[1] over the past ten years [49]. In 1993, only 1,334 incidents were reported, and this number grew to 3,734 in 1998. In 2003, however, the number was an astonishing 137,529, which is more than triple the total number of incidents reported since 1988. During this time period, intrusion detection systems and firewalls have become more prominently used to prevent or mitigate such attacks. Suffice it to say, proactive security measures are not always adequate for preventing every attack.

Since computer attacks cannot be fully prevented, a response strategy must be in place. A computer attack must be responded to in a quick and effective manner. With every attack comes the threats of data corruption, backdoors, and the possibility of the system being used for distributed denial-of-service attacks or as a server for illegal files. The simplest, and initially most appealing, solution is to format the compromised system and reinstall its software. This response removes all possible malicious code left on the system and is fast – two compelling advantages. But this response also removes all traces of information that could lead to an understanding of who committed the attack and how the

[1]The Computer Emergency Response Team Coordination Center (CERT/CC) is a federally-funded group at Carnegie Mellon's Software Engineering Department. They analyze and publicize incidents reported by computer administrators internationally.

attack occurred. In addition, any system data that was not backed up prior to the incident will be lost.

The disadvantages inherent to the abovementioned response underscore the need for conducting a post-incident investigation. The investigation should note and collect all important data and then carry out an investigation before any changes to the system, such as reinstallations, occur. Such an investigation may not be as quick or simple to perform as the previous response, but a more complete solution offers better long-term benefits. These benefits include an understanding if how the attacker obtained access to the system and of how to prevent such attacks in the future, as well as discovering the identity of the attacker and possibly pursuing legal action. The area of security that is dedicated to responding to an incident is known as computer forensics.

This chapter defines computer forensics and presents this thesis' problem statement. Computer forensics is a misunderstood field that is multi-faceted and still developing. These issues are considered in the first section. The second section discusses the problems with current computer forensics analysis methodologies. Since computer forensics is such a new field, its body of literature is still small, and the methodologies have not been satisfactorily formalized.

1.1 Computer Forensics Defined

The College of American Pathologists defines forensics as "the application of the principles of the physical sciences in the search for truth in civil, criminal, and social behavioral matters to the end that injustices shall not be done to any member of society" [16]. Forensics is the collection and analysis of physical evidence directly relating to the investigation and all external evidence that may impact the analysis. A large body of literature has established precise methodologies and techniques for proper handling and analysis of evidence. This literature, however, has not included computer forensics, which is replete with its own idiosyncrasies and requirements.

Computer forensics[2] is broadly defined as "the collection, preservation, analysis, and presentation of computer-related evidence," [50]. Using this definition, forensics can be applied to any situation that requires some data from a computer to be collected and

[2]For the remainder of this paper, the term "forensics" is used to denote "computer forensics."

examined. Some mistakenly call forensics "data recovery." Although forensics does involve data recovery, the process of performing forensics is much more complex than simple data recovery. Once data has been recovered, the investigator needs to piece together the evidence in a coherent manner, not simply recover deleted files or find hidden files.

The scope of this definition does not necessarily exclude non-data evidence; there may be cases in which a user's past actions or knowledge may be useful in solving the causes or effects of an incident. For example, a system administrator may have noticed anomalous network activity on the network, but before he reported the incident, he attempted to mitigate the problem by removing several suspicious files on a server and blocking several network ports. These actions play a role in the forensic investigation. Moreover, if the administrator had previously antagonized a person capable of carrying out such an attack, the investigator could use that non-data information as a clue for determining the source of the incident.

Several salient aspects form the core of forensics in practice. First, all data are important, including volatile data. An investigation must thus be complete with respect to collecting all pertinent data. Second, the analysis of the collected evidence must proceed in a structured manner. The structure can be determined either by the nature of that particular incident or by a generalized plan. For example, an investigation of an end-user's compromised Linux computer should follow a generalized plan for all end-user Linux computers, which would consist of looking for signs of a known exploit or other vulnerable points within the Linux operating system. A corporate back-up system should be investigated in a more case-specific manner. This case's analysis should consist of performing risk analysis of the computer and proceeding to analyze those areas that are at a higher risk of being compromised or failing. Finally, forensics consists of multiple phases. These phases should be followed in pre-set order so that the investigation is more efficient and less prone to errors.

Understanding computer forensics requires knowing about the core set of phases in a forensic investigation. The basic set of phases includes the collection, recovery, analysis, and presentation of evidence. Researchers and practitioners have established these as the basic set of phases. There are several variants of the phases, but all of these variants are the result of subdividing the phases in different ways [50], [30], and [26]. All of the methodologies perform the same four phases in a similar manner; each allows a system to be investigated in order to understand what caused and happened after the incident, fix the damage done

to the system, and prevent future such incidents. This thesis uses the four major phases without any of the subdivisions. These phases are preparation, collection of evidence, the analysis of evidence, and post-analysis action.[3] Together, these phases provide a complete set of phases that can be used to produce a thorough investigation.

An investigation that follows the abovementioned phases begins when a system behaves abnormally, e.g. an intrusion detection system (IDS) issues an alert or network bandwidth is high. An investigatory plan is formed on the basis of the nature of the system behavior and the results that are desired from the investigation. Next, all pertinent data is collected from the system, which includes volatile data, logs, and possibly an image of the file system. The collected data is then analyzed to determine the cause(s) of the incident. Finally, the analysis is used to establish what actions should be taken to recover from the incident. Potential responses include system patching, restoring system backups, or even legal action.

1.1.1 Types of Computer Forensic Investigations

The type of forensics involving the process of responding to computer attacks is known as incident response. The goals of incident response are to quickly determine the cause of the incident, mitigate the negative effects, and prevent similar attacks from reoccurring. Incident response is usually time-critical and places a high emphasis on mitigation, recovery, and prevention. An investigation may involve determining source of the attack and pursuing legal actions against the attacker. This goal requires that the investigator take great care when conducting the investigation to ensure the evidence's court-admissibility so that the evidence cannot be refuted by a defense attorney.

While a large body of research has focused on forensics as a way for responding to computer attacks – that is, incident response, forensics is not limited to this type of investigation. There are numerous other applications of these techniques. Law enforcement, for instance, uses forensics for the investigation of illegal activities in which a computer was used to commit or store information about an illegal activity. A significant number of crimes are now being conducted with the aid of a computer. The crimes range from credit card fraud to identity theft to terrorism. Law enforcement investigators are not concerned with returning the computer to a running state; they are only concerned with collecting as much

[3] See Section 2.2 for a thorough discussion of these phases.

incriminating, court-admissible evidence as possible. Two recent high-profile examples of such investigations include the Enron scandal [25] and the Zacarias Moussaoui case [2].

A third application of forensics is the diagnosis of unreliable or otherwise faulty system. A common form of troubleshooting a system is an *ad hoc* process by which a system is analyzed for anomalous or incorrect behavior. The system is then reconfigured and tested until the computer works correctly. Rather than applying these *ad hoc* techniques, forensic analysis may be applied. Forensic techniques can be used to accurately diagnose problems, and that forensic analysis is then applied to form a more effective solution than that of the *ad hoc* process.

1.1.2 Terminology

Used throughout this thesis are a number of terms with forensics-specific definitions:

- Incident: a chain of events related to one another in purpose or effect. A network-based denial-of-service attack, with the chain of events including a probe of the victim network and a set of packets sent to the victim that caused the denial-of-service, is an example of an incident.
- Investigation: the performance of the complete set of forensics phases, including the preparation, data collection, analysis, and post-analysis actions.
- Investigation requirements: the collection of investigation priorities. The two main requirements are completeness and timeliness.
- Data: any information that may be collected and analyzed for an investigation, which includes both electronic data and non-electronic empirical data.
- Evidence: a logical grouping of data, forming a single clue. The logical grouping ranges from extremely low-level data, such as a single sector on a hard drive; to mid-level data, such as a network packet or a small text file; to extremely high-level data, such as an entire file system
- Methodology: a particular procedure or set of procedures that are applied using a set of known techniques.

- *Ad hoc* analysis approach: the absence of a particular procedure or set of procedures for forensic analysis, whereby the investigator must rely on his own judgments and intuition when performing analysis.

1.1.3 Jurisprudence Considerations

The three types of forensic investigations – incident response, computer examination, and troubleshooting – illustrate forensics' combination of technical analysis and elements of jurisprudence. The jurisprudence aspect is a topical issue that affects both law enforcement and independent forensics practitioners, since most investigations do involve some illegal activity that necessitated the investigation. Questions regarding what constitutes verified evidence and how an attack can be completely traced back to the attacker's computer are difficult and sure to be examined by law experts. This thesis, however, does not consider such legal issues. The focus of this thesis is to examine the structural organization of the analysis phase and to provide a better methodology for an investigator. While the methodology certainly appears to be adaptable for law enforcement purposes, the focus is on the strictly technical matter of how analysis can be structured in a more coherent and logical manner.

1.2 Problem Statement

Current forensics research focuses on three major issues: managing the audit reduction problem, quickly detecting anti-forensics deception, and semi-automating data collection and analysis. The audit reduction problem is trade-off between completeness and timeliness. An investigator must manage a large bulk of collected data while paring down the amount of analysis that must be performed. An investigation should be completed in a timely manner; however, the desire for timeliness can threaten completness, as decreasing the amount of time spent on analysis also decreases the amount of evidence. This trade-off is a problem inveherent to all investigations. The problem is typically resolved by using case-specific knowledge of a system to determine what information is critical, thereby minimizing the amount of extraneous information that is collected and analyzed, and guaranteeing that no critical data is omitted.

The second issue, anti-forensics, is an ongoing battle between investigators and hackers. Anti-forensics is a term used to describe any intentional means of hiding or modifying data so as to thwart a proper forensics investigation. A hacker or other malicious user performs anti-forensics techniques to conceal his actions and presence. These techniques include stenography, modifying logs, and the use of root kits. Researchers have attempted to combat these problems by using steganalysis, checksums to test for trojans, and timeline analysis on multiple logs to test for log tampering. While investigators are now better able to cope with anti-forensics techniques, a skilled hacker still has the advantage due to increasingly sophisticated anti-forensics efforts [47].

Finally, research into semi-automation of forensic investigations uses collections of tools to perform tedious tasks that the investigator would otherwise have to do himself. The benefits of automation include the expedition of an investigation and the prevention of mistakes, e.g. mistyping or not performing a routine task. The use of tools to semi-automate an investigation is still limited. While the data collection phase is almost fully automatable, the analysis phase only allows small tasks, such as checking for rootkits or making a timeline from the system logs, to be analyzed. The process by which a conclusion is reached is currently not fully automatable, though it may become so in the future.

One notable effort to automate the analysis process uses the notion of invariants to establish a set of conditions by which the automated analysis can be conducted. The invariants are determined by a set of conditions that are established at the beginning of the analysis. The remaining analysis makes use of the invariants to determine the validity of the evidence. This ambitious attempt at automated analysis is only demonstrated for trivial cases. Chapter 2 examines this type of analysis and shows why it is cumbersome and error-prone, despite its significant aims.

Research in this field approaches the issues surrounding forensics as if the issues were a dichotomy. The researchers' focus tends to divorce high-level approach issues - such as the preparation and investigatory phases - from low-level technical issues. For example, researchers have worked towards a method for quickly and accurately determining whether system processes have been altered by an attacker. This technical matter is important, but nowhere in the research literature is there any mention of how to incorporate this information into an investigatory plan. Researchers have not integrated such low-level technical issues into a high-level investigatory plan, leaving that matter for the investigator to determine.

The aforementioned false dichotomy is further illustrated by the way in which the investigatory phases are presented in forensics literature. The phases are presented as a list of steps that are performed linearly, starting with preparation and ending with post-investigation actions. This linear ordering of steps is analogous to the waterfall model used in software engineering [37]. In the waterfall model, the coding process begins with the design phase, proceeds with analysis and coding phases, and ends with the coding and testing phases. The model is simple and intuitive, but it does not allow for backtracking to previous steps. The limitation proves troublesome when a mistake with a previous phase is discovered during a later phase. The model does not directly support reiterating through the steps, but since reiteration is required to correct mistakes, the model must be adjusted to allow for the mistakes to be corrected. Adjusting the model during a software project can cause confusion, which leads to further mistakes and delays [21].

The problem with the forensics model of phases is that, like software engineering, investigations are dynamic and are often not linear. Investigations begin with a set of clues that provide the investigator with ideas for determining the cause of the incident. The investigator collects as much data as possible, but some evidence may not be collected due to being viewed as extraneous. When the analysis begins, the initial clues serve as leads, allowing the investigator to form a general plan. The plan, however, may lead to a dead-end in which the evidence is either insufficient for proving the presupposed cause or indicates a different cause. In either case, the investigator must, as it were, take a step back, reformulate the plan and begin analyzing again. Reformulating the plan requires reiterating certain steps and phases that have already occurred. This linear model is insufficient for most forensic investigations in the same way as the waterfall model is insufficient for most software engineering projects.

The forensics model of phases can be modified to allow for iterations of the phases by using an iterative model. Because data collection is typically performed only once, the phases that are generally performed multiple times are the preparation and analysis phases. Considering the frequency of the analysis and preparation phases, an iterative model can be conceived. The first type of iteration occurs when an investigation plan needs to be modified slightly within the analysis phase. This type of modification is called for when an investigatory plan proves useful, but analysis shows that it is lacking in minor ways. For example, if an investigator who is focusing on a network backdoor discovers that the backdoor could have been planted via an opened email attachment, the investigatory plan

should append email to the list of data to be examined. However, if an investigatory plan produces little or no results, it should be discarded, and the investigator should go back to the preparation phase to devise a new initial plan. After forming the new plan, the investigator forgoes recollecting data and proceeds to the analysis phase again. The iterative model is shown in Figure 1.

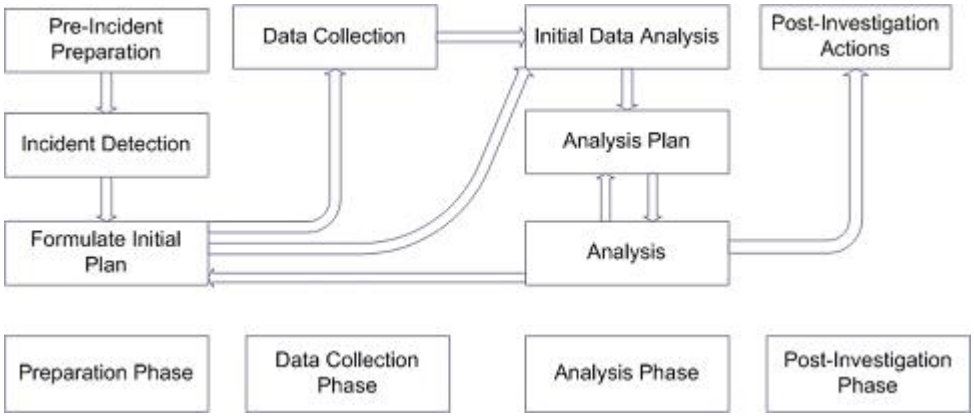

Figure 1.1: An example of an incremental investigation model.

The iterative model is more representative of an investigation than the linear model, but it introduces several difficulties of its own. First, the model does not present a means for systematically incorporating previous analysis into the formation of a new or modified plan. The aim of every phase is to complete the investigation without performing another iteration. Because of this all-or-nothing approach for each phase, incorporating past analysis into a new plan is non-trivial. Second, the model does not provide any means for melding the high-level approach of the analysis phase with the low-level details that comprise the bulk of that phase. Combining set of low-level details into higher-level, more abstract information is how investigators perform investigations. Instead, the analysis phase is viewed as a static process in which low-level details are collected and then analyzed via *ad hoc* means, while at the same time following the original plan.

The problems facing the iterative model have not yet addressed by forensics researchers. The main failing is that researchers have not approached forensics as a dynamic process involving a wide range of details. Instead, they tacitly presuppose a false dichotomy

by analyzing only high-level and low-level details. This false dichotomy leads to confusion and inefficiency when an initial plan has to be modified during the analysis phase. Instead of using a known technique, investigators must reformulate the plan in an *ad hoc* manner. This is problematic because of the confusion and convolution that results from reformulating plans, since the investigator must properly and accurately do so while considering the audit reduction problem.

The bulk of every forensic investigation is the analysis phase. It is in this phase that an investigator must comb through data, analyze them, and formulate ideas about their inter-relationships. The preparation phase is largely an administrative task involving the understanding of what took place and applying known standards for beginning an investigation. The data collection phase applies the rules from the preparation phase and can largely be automated, so investigators are not troubled by that phase. Finally, the post-analysis phase is typically the application of the information taken from the analysis to correct what occurred. Analysis requires far more time and mental effort than any of these phases. Research should therefore focus on establishing a strong foundational methodology upon which analysis can be performed.

1.2.1 Computer Forensics Literature Problem Statement

Computer forensics research has not completely ignored the problem of lack of a rigorous methodology for the analysis phase. Several researchers have noted this problem and have established a set of criteria by which a new methodology should be based. Gary L. Palmer claims that computer forensic analysis is not a true science, since the established methodologies are based on reactions to practical needs, rather than based proactively on sound scientific principles [33]. The confusion exists because of several misconceptions as well. First, traditional forensic science is founded on sound scientific principles from the soft science of biology and the hard science of physics. The principles and methodologies of traditional forensic science have been thoroughly scrutinized over the past hundred years in both academic and legal circles. During this time, both sides realized that no evidence is irrefutable, and that the ultimate conclusion must be convincing when faced with counterexamples or questions of the soundness of the analysis. The scrutiny has led to a well-accepted body of literature. Computer forensics, however, is a relatively new field. Computer crimes began to occur before technical and legal researchers truly understood

the underlying problems of analysis methods and the validity of such analysis. This lack of understanding has resulted in a series of methodologies that are designed to analyze specific types of cases. For example, many of the analysis techniques and methodologies designed by practitioners are meant to handle hacking cases, whereas those designed by government agencies are meant to handle cases involving computers being used as an accessory for a crime. Palmer notes that the scientific community has been absent from the methodology formation process, since most of the work is being done by practitioners and government agencies.

A second problem that Palmer discusses is the misconception of certainty in forensic investigations. Computers are mathematical machines that provide accurate representations of data. An investigator should be able to query computers in such a way that the full cause of an incident can be understood. This, however, is mistaken when humans are involved. A sophisticated suspect is able to alter evidence in such a way that the analysis does not produce irrefutable analysis. Moreover, modern computer systems, with their inter-networking and complex operating systems, are so complicated that an investigator can have difficulty uncovering and correlating all of evidence in an irrefutable manner.

Future analysis methodologies must meet several criteria and be tested in controlled environments. The first requirement is that the methodology must be able to produce relevant and/or material evidence. That is, the evidence must be valid with respect to the analysis conclusions. Second, the methodology must produce credible and/or competent evidence. The methodology must be justifiable in its approach so that believable, trustworthy, and true analysis is produced. These criteria are the same upon which traditional forensics is based. In addition, to support the soundness and validity of the methodology, it must be tested in controlled environments and be subject to the scrutiny of academia, practitioners, and jurisprudence researchers alike. Only in this manner can the methodology be scientifically sound.

An examination of current analysis methodologies by Reith et al. shows that no current methodology meets these scientific requirements [12]. The authors explain that the majority of practitioner-developed methodologies are too narrowly focused and are not generally applicable. A basic methodology put forth by Dan Farmer and Wietse Venema in 1999 focuses on investigating Unix operating systems [13]. Farmer and Venema gave a series of lectures on performing computer forensics for the Unix operating system in which they discussed the basic principles and techniques for investigating incidents that were specific

to Unix. In addition, they also released one of the first forensic tools designed to aid investigators with an investigation of the Unix file system [14]. While the techniques and methodology are suitable for Unix, they do not apply to other operating systems or types of cases. One problem is that many variants of Unix exists, so the methodology may be acceptable for Solaris, but it may not be suitable for Linux, HP-UX, or BSD. In addition, operating systems constantly add and remove features; this requires the investigator to alter the methodology based on which operating system release is being investigated. As such, the methodology is questionable and is certainly not appropriate to be a model for computer forensics.

Another type of model, the U.S. Department of Justice's (DOJ) model, is also considered by Reith et al. The DOJ base their model on the types of evidence and media being examined. They propose different techniques for each, which is then cross-checked with a list of crimes to determine the type of incident that occurred. Once again, this model is designed to solve the immediate problem of investigating crimes that are specific to a point in time. In 1995, for example, credit card fraud comprised a large portion of computer crimes. In 2004, however, identity theft is a higher concern. The types of crimes that occur in 2010 are yet unknown, so the methodology must be updated as new crimes become prevalent. The DOJ's methodology is thus too specific to a certain set of crimes to be considered a general, scientific methodology.

1.2.2 Proposed Solution

The shortcomings of existing methodologies can be solved by formulating an analysis methodology that is independent from existing technologies and incidents. Instead of focusing on how to solve a specific problem in a specific scenario, the methodology should be designed to handle evidence in a logical fashion with respect to the investigator and those to whom he is reporting his findings. This requirement means that the methodology should focus on the practice of analyzing digital evidence in general. The methodology should produce valid results in a sound manner. The technical issues of conducting the investigation in a specific scenario with a particular set of technologies being involved are left to the investigator. An investigator dealing with these technical issues is often aided by analysis tools and a set of guidelines for that investigation, so the methodology need not encompass those problems.

This thesis seeks to bridge the gap between the high-level and low-level aspects of investigations by proposing a formalized analysis methodology. The idea behind the methodology is that forensic analysis can be viewed as an inductive argument. The hypothesis of this argument is the believed causes and effects of the incident. The goal of an investigation is to produce convincing evidence, much like a biologist does. The investigation thus proceeds in a manner much like that in the natural sciences. To prove the argument, a series of premises must be satisfied. These premises are typically conclusions to smaller, more technical arguments. The low-level arguments are either inductive or deductive, but the goal of each is the same: to provide a bridge between the low-level details and the high-level argument.

Using this thesis' methodology, an investigator begins an investigation by outlining the possible causes of an incident based on his understanding of the system and the incident. After performing the data collection phase, the investigator forms a general supposition about the cause of the incident, called the macrocosm hypothesis. Next, the various clues that suggest the cause are analyzed. These clues serve as premises for the macrocosm hypothesis, which is the overall hypothesis for the entire analysis. To prove that the clues do entail the macrocosm hypothesis, each must be proven by analyzing the low-level details. Proving these low-level details is achieved by using one of three types of arguments: high-level pattern matching (deduction), low-level pattern matching (deduction), or microcosm hypothesis (induction). As the investigation proceeds and premises are accumulated, the cause of the incident shall become clearer. The macrocosm hypothesis shall either be proven or disproved after enough of the premises are proven.

Using syllogistic logic to form proofs for an investigation seems to be a rather intuitive and simple task, but actually performing an investigation using this methodology is a different matter. One problem is that an incident is complex, and expressing an incident using just a single hypothesis is difficult. Another problem is how to reformulate a plan using this methodology. If the overall plan fails, then the investigator must reformulate the plan, but the previous analysis is already expressed with respect to the previous hypothesis. The final problem is constructing the premises to prove the hypothesis; the investigator does not have total *a priori* knowledge of the system and the incident, so there should be guidelines for constructing them.

This thesis presents the methodology in a theoretically complete manner by drawing from other data mining research areas, such as intrusion detection and artificial intel-

ligence, while demonstrating the practical applicability of the methodology. Many of the problems that face forensic analysis, such as plan reformulation and mid-level data representations, have been addressed in the literature from other fields. The logical techniques and methodology in this thesis draw from those other fields to solve the aforementioned problems. Moreover, the methodology is similar enough to the previous forensic investigation phases model that it retains the practical usability.

1.3 Thesis Format

This thesis is divided into five chapters and one appendix. The first chapter defines computer forensics and its various applications, and the problem of current computer forensic analysis methodologies is presented. Chapter two discusses the current techniques used for forensic analysis, the problems associated with *ad hoc* analysis, and several attempts at formal analysis methodologies. The third chapter presents the formalized methodology and techniques for a complete forensic investigation; this chapter provides the bulk of the theoretical contributions of this paper. The focuses are on the logical structure of the techniques and how they applied. The fourth chapter is a collection of four forensic investigations that are performed with this thesis' formalized analysis methodology; this chapter demonstrates the practical value of the methodology and its limitations. This chapter also juxtaposes the methodology against previous *ad hoc* analysis. The final chapter contains concluding remarks and future work to extend this subject. Appendix A provides the paper, entitled "Cross-Validation of File System Layers for Computer Forensics," which was presented at the Digital Forensics Research Workshop in 2003 by the author.

Chapter 2

Current Computer Forensic Analysis Techniques

Forensics analysis is a dynamic process in which an investigator logically pieces together the causes of a security incident and determines its effects. Analysis begins with of general knowledge about the system and the incident to formulate a general plan for analyzing the system, and then to apply that plan using the data collected during the data collection phase. The analysis itself consists of a series of techniques that exposes possible causes. The three main techniques are common vulnerability analysis, timeline analysis, and baseline analysis. There are other techniques - such as cross-validating log entries - that may be used, but the logic behind the other techniques are the same as the three major ones. The investigator applies these techniques as he deems necessary, for the literature provides no guidelines as to which technique is most appropriate in a given circumstance.

The analysis phase is not a linear process that follows a single plan; it is a dynamic process involving hypothesizing about causes and effects and correlating known information with other data. Analysis typically occurs after an investigatory plan has been formed and all pertinent data has been collected, at which point the investigator examines the data to confirm or refute suspected causes. The investigatory plan, however, may not be accurate or true, so the investigator must return to the preparation phase and devise a new plan. This

process continues until the correct plan is formed and the investigator is able to accurately determine the causes and effects of an incident.

The analysis phase cannot be thought of as independent of other phases; all of the phases are in some way intertwined. A typical investigation proceeds in a similar fashion to the scenario in the paragraph process: a plan is formulated, the plan is used but does not help the investigator solve the incident, the plan is reformulated, and the previous steps are repeated until the correct plan is used to finish the analysis. Observing that analysis can require returning to the preparation or data collection phases is important for understanding the complex nature of analysis. An investigator does not have full *a priori* knowledge of what a specific investigation will entail, so he must adapt his strategy as new information about the incident is gathered. This adaptation is facilitated by returning to previous phases and making the appropriate corrections before resuming the analysis.

The main difficulty with this method of analysis is not the intuition behind returning to previous phases, but rather the unstructured way in which the phases are intertwined. The two circumstances in which an investigator leaves the analysis phase to return to a previous phase are when the investigatory plan is shown to be a dead end that produces no results and when important new information about possible causes is discovered during analysis. In both cases, the investigator must return to a previous phase, and using what he learned from previous analysis, reformulate the plan. Current forensics literature provides no specification or best practice standards for integrating previous analysis when forming the new plan, and it does not discuss the circumstances at which analysis should give way to a plan reformation. The investigator is thus left to his own devices to correctly form a plan that makes full use of past analysis in an efficient manner.

This chapter details the current state of forensic analysis and how it relates to the three other phases. The first section outlines each of the four investigatory phases and the requirements of each; this section is important for understanding how the analysis phase relates to the investigation as a whole. The second section discusses the three major analysis techniques: baseline analysis, common vulnerability analysis, and timeline analysis. This section demonstrates the common techniques employed by investigators. The third section analyzes three analysis methodologies. The first methodology is post-incident root cause analysis, which models analysis using Colored Petri Nets. The second methodology uses the notion of evidence invariants to conduct the analysis. Finally, *ad hoc* analysis is discussed. Since much of the uncertainty and inefficiency associated with analysis is due to the *ad hoc*

approach, the reasons for the lack of a rigorous, standardized methodology are discussed here.

2.1 Computer Forensics Phases

The four major phases form the skeleton of forensic investigations. These phases are part of any forensic investigation, so one must understand each of the phases and how they are related to one another. This section gives a detailed account of the phases, including the common steps that comprise each.

2.1.1 Preparation

An investigator must be fully prepared to begin a forensic investigation before the need for such an investigation arises. The preparation phase requires the investigator to prepare both a specified investigation policy that details all procedures to be followed and a set of software tools called a forensic toolkit [40]. The first step is to form an investigation policy that outlines what information needs to be collected and how to collect it. Next, a set of forensic software tools and hardware to be used for investigations, called a toolkit, should be assembled.

Investigation Policy

The investigation policy details the general approach that future investigations should take. It combines the features of the system(s) to be investigated with the requirements of the type[1] of investigation. The two requirements for this policy are that it is thorough with respect to the investigation requirements and that type is more of a malleable template rather than a rigid checklist.

Thoroughness, the first requirement of an investigation policy, verifies that a policy is complete in terms of investigation requirements. In this context, "complete" refers to the extent to which the investigation follows the best practices for investigations and

[1] "Type" denotes a specific case. For example, incident response for a hacked end-user computer is an investigation type.

utilizes the foremost technical resources. The term does not and cannot be used in an absolute, exhaustive sense, and it also applies to vulnerability analysis and intrusion detection. An investigation policy makes use of the best standards and technical resources that are available.

Technical resources pertaining to security are popular choices for creating investigation policies. A combination of print and electronic sources can help the investigator arrive at a sound investigative policy. Websites supplement books by providing the low-level details of vulnerabilities and attacks that most books lack. Together, books and websites provide a means for understanding which aspects of a system are commonly problematic.

Perhaps the most valuable, yet most frequently overlooked, resource for forming an investigative policy is system documentation [54]. Knowledge of system internals - especially what constitutes normal system behavior - is critical for conducting a proper investigation. System documentation is often the most thorough description of system internals. The system documentation can be used as an exhaustive list of applications, services, system configuration, and system logs. Additionally, the documentation provides information about system nuances, and nuances that, in turn, may be used to formulate additional policies. For example, the documentation may discuss how to recover deleted files and where slack space[2] exists on the system. The policy could then include steps that check for deleted files and analyze system slack space.

System Baseline

System baselining is an important proactive measure for forensic investigations that gives reference points to determine which system components have been altered [12]. A baseline is established by taking a snapshot of important configurations, checksums, and system processes. The system baseline contains a wealth of valuable information for investigations that quickly points to the cause of an incident when compared to the snapshot of the system taken during investigation. The clues given by a baseline include file access and modification times, drastic increases in certain system resources, and identification of rogue system modules.

Ideally, baselining is performed in regular intervals on the actual system being

[2]Slack space refers to areas within files or file systems that may remain unused or may be used as storage for hidden data.

used. This method allows the baseline to reflect recent configurations and system changes. Less effective techniques include performing a single baselining of the actual system and baselining the factory media without the configurations used for the implemented system. These techniques yield some insight into how a normal system appears. However, most running systems have idiosyncratic configurations that make comparisons to factory media or original configurations too disparate, so the comparisons between the two are not useful.

Creating a useful baseline for forensics is summarized as follows [32]:

1. Define the system attributes and components that are important to a forensic investigation.
2. Run a minimally invasive tool to collect and archive the attributes and components listed in Step 1 in a format that is suitable for comparison to a future system snapshot.
3. Periodically update the system snapshot by repeating Steps 1 and 2.

Toolkit

A forensic toolkit is a collection of software used for investigations. An integral part of the investigation, it can enhance the speed and effectiveness of data collection and analysis. The toolkit needs to be tailored to the type of system in question and the forensic policy. Most investigations, however, follow a procedure similar to those of other investigations' procedures, so toolkits often have a set of core tools. These include scripts to automate data collection and analysis, disk imaging tools, and trusted binaries of system tools (in case of trojanized system tools).

While the actual tools chosen are investigation-specific, there are several guidelines that should be followed when constructing a toolkit. The toolkit should perform via automation as much rote data collection and analysis as possible. Automation reduces the likelihood of human errors like forgetting to gather some evidence or mistyping a damaging command, and increases the speed of an investigation. The toolkit could, for example, contain a simple script that collects the contents of all volatile data, thereby increasing the likelihood that data is not lost while recovering it.

Analysis should be performed in a trusted shell, if possible, and only trusted binaries should be executed. The toolkit must contain these items. A system may be

compromised by a hacker or a process on the system may be unstable; an investigation should never be performed under a contrary assumption. Running trusted binaries also eliminates the problem of older, less robust applications that may be on the system.

Preliminary Examination

All investigations begin with the occurrence of one or more anomalous system behaviors occurring. These clues are what prompt an investigation; without any clues, there is no reason to investigate. Before an investigation begins, the initial clues, system information, and investigation policy should be used to form an investigation plan. The plan aids the investigation by providing hypotheses, or at the very least, a starting point for analysis.

Merging clues and system details with an investigation policy is a dynamic and *ad hoc* process. A typical scenario involves a system displaying a single anomalous behavior - for instance, drastically increased network traffic. Since the investigation must determine the cause of the behavior in order to proceed, the plan should emphasize that all network evidence related to this behavior is to be gathered. The volatile network data, moreover, should be given higher priority than equally volatile system data.

The investigation policy forms the basis of the investigation plan, whereas the clues make alterations to the plan given by the policy. The clues should not cause the investigation plan to depart drastically from the investigation policy. The clues should only provide direction for the investigation and emphasis on certain types of evidence; the overall investigation policy should comprise the bulk of the investigation plan. The clues are valuable insofar as they offer insight into an incident. Evidence that is not directly related to the anomalous system behavior should still be collected as it may need to be analyzed at some later time.

2.1.2 Data Collection

The two goals of data collection are to maximize the usefulness of the evidence and to minimize the cost of collecting it [46]. These goals are not mutually exclusive; in fact, they overlap quite frequently. Increasing the usefulness of evidence typically involves collecting corroborating evidence, which increases the data collection cost. The balance

between evidence cost and usefulness must be based on the overall goals of the investigation.

An investigation should be performed quickly so that the affected system can be returned to its regular job. The presumption is that quicker data collection results in a quicker investigation. However, this call for speed can result in the collection of insufficient evidence, the worst-case scenario for an investigation. Insufficient evidence means that data collection, a time-consuming process, must be performed again. Even worse, the needed data may be volatile and thereby lost in the first data collection due to system usage.

The primary goal of most investigations is thoroughness, since the main purpose of an investigation is to arrive at a complete understanding of the causes and effects of an incident. The data collection should thus be carefully performed so as not to overlook data that could potentially be useful, even if the data does not directly relate to any of the clues collected from the preparation phase. The standard method for ensuring that all potentially useful data is collected is to gather all volatile data, logs, and other system-specific information that may not be gatherable at a later time. Collecting this information allows an investigator to begin investigation without the possibility of losing potentially useful information.

Imaging, the process of creating a complete copy of disk drives, is a frequently performed data collection practice. An investigator may wish to make a complete sector-by-sector copy of disk drives so that he can do low-level analysis at a later time. Simply copying all of the potentially useful files from a system may not be sufficient for a complete investigation. For example, if a hacker were to delete all file system-level clues, the investigator would not be able to fully piece together the incident. Making a complete image of a disk drive allows the investigator to search the hard drive for deallocated sectors that contain useful information. Most operating systems do not overwrite the contents of sectors upon deletion; instead, the sectors are simply unlinked from the file system table and left untouched until the operating system requires use of that sector. Analyzing the disk image therefore allows the investigator to delve deeper into the system's state and perform a more thorough investigation.

Guidelines

Data collection is a mechanical process governed by strict guidelines. These guidelines must be established before collection begins. One should not adapt when collecting

data unless the system behaves unexpectedly during the process; spontaneous creativity is not advantageous in data collection. The investigation plan should thus be followed unless circumstances require that a back-up data collection plan be followed.

Toolkits automate the data collection process by using scripts and data collection software wherever possible. Automation provides the advantages of faster collection and error-free execution of the procedure. The scripts reduce the amount of typing required, which means that no user typing errors will occur. The delays associated with typing numerous commands are also eliminated.

The investigator should image all disk drives unless circumstances deem otherwise. The cost of imaging is not great. The investigator uses a sector-copying tool, such as *dd* or *EnCase*, to send the image to either an attached investigation hard drive or to a forensic workstation system through a network connection. The process' cost is determined by the speed of the hard drive or the speed of the network connection. The only cases in which it is not prudent to image every disk drive are when an incident is trivial to solve or the amount of data to image is too great.

Order of Data Collection

The problem of permanently losing data is avoided by guaranteeing that all volatile and temporary data is collected first. Volatile data are types of memory that are highly sensitive to system usage, e.g. registers, memory, and cache. Such data are lost whenever a system is used, so they should be collected first to minimize corruption or loss. Additionally, all temporary data should be collected after the volatile data but before persistent data. The likelihood of corrupting temporary data is less than that of volatile data, but temporary data is just that - temporary - and it must be collected before it is lost. The order in which data should be collected is given in Figure 1.

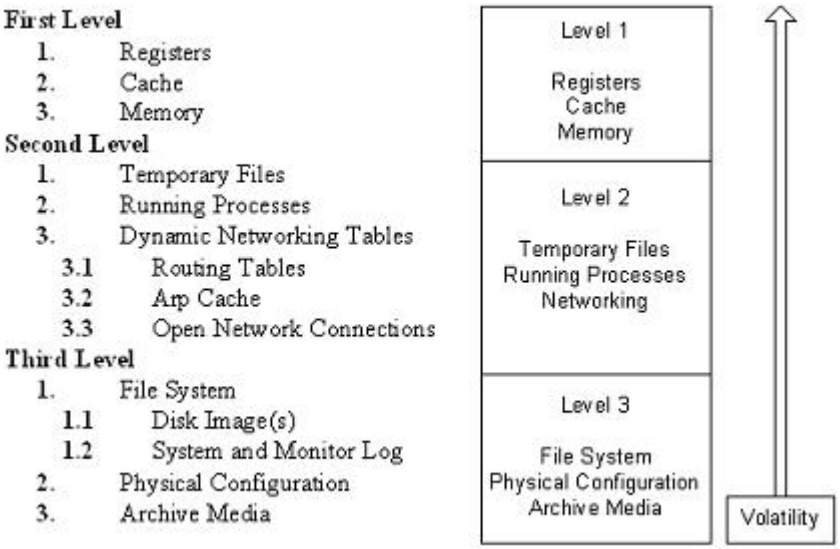

Figure 2.1: The three-tiered data collection model.

2.1.3 Analysis

Analysis is the heart of forensics; the other three phases are either preparations for analysis or applications of the analysis. Analysis is also the most complex phase, whereas the other steps can be partially automated and performed mechanically, analysis combines mechanical elements with a sense of intuition and creativity. Because of its importance and complexity, this phase should be given the most consideration and effort.

The analysis phase can be divided into three phases: initial data analysis, investigative plan formulation, and analysis. The first step, initial data analysis, involves checking the critical data that most closely corresponds to the plan formed in the preparation plan. The goal of this step is to quickly confirm or refute the suspicion about what caused the incident. If the initial data strongly supports the initial suspicion, then no changes need to be made to plan. If, however, the initial data either does not support the suspicion or supports another possible cause, then the plan needs to be modified. The second step, reformulating the investigatory plan, is performed if the initial data analysis or later analysis does not strongly support the original investigatory plan. This step is performed by the investigator

when he needs only to make minor adjustments to the plan. Here, he only makes use of the prior plan and the initial data analysis. The investigative plan formulation step is the simple process of comparing the initial data and the investigative plan with the preparation phase. If the two are consistent, then no changes are made to the plan. If the initial data suggests a plan that differs slightly from the direction of the initial plan, then the investigator makes the slight adjustments and begins analysis. Otherwise, the investigator must reformulate the plan and possibly collect additional data.

The analysis itself is the final step, of which the goal is to examine all relevant information and correctly determine the cause and effects of an incident. This step begins by applying the investigative plan and an initial clue to the collected data. The initial clue is examined, and the investigator then makes a decision based on the clue, the system specifications, and the forensic plan about the next clue to pursue. This process of clue examination is an iterative process that continues as long as no evidence is discovered that refutes the investigatory plan. If the investigator examines enough confirming evidence such that he is confident in the result of the analysis, then the analysis phase is concluded. If, however, the investigator discovers refuting evidence, he must reformulate the investigatory plan. The reformulation occurs either in the analysis phase or the preparation phase, depending on the severity of the refutation.

The main difficulty facing this step is the need to minimize the amount of extraneous data that is analyzed. The investigatory plan, however, acts as a checklist for reviewing the critical information. For example, if an incident caused an intrusion detection system to send numerous alerts at a certain time, then system logs can be reviewed to find anomalous behavior that occurred around that time. Further information may be in these logs which provides more information, such as failed login attempts, about the nature of the attack. This type of correlating and cross-validating is the standard way in which an investigation develops. The amount of data to be analyzed is thus minimized by following a chain of related information rather than analyzing all possible information. The analysis can be visually represented as a path through a maze - albeit with several wrong paths taken during the process - instead of a complete connected graph of information. Below is a depiction of an example analysis path in which data analysis follows.

In Figure 2, E1 is the piece of evidence that is the starting point of the analysis, the first clue analyzed. E1 leads to E3, the next piece of evidence. The path then goes from E3 to E2 to E6. At this point, the evidence E10 looks promising, but it does not provide

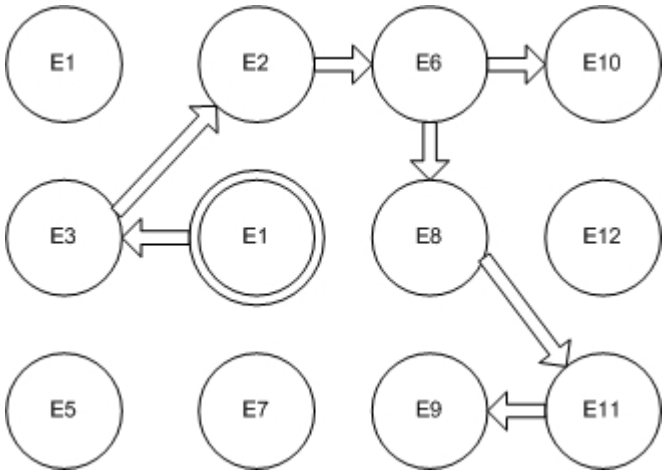

Figure 2.2: A representation of an example analysis path.

any useful information. So E6 leads to the next useful piece of information, E8. E8 leads to E11, and E11 leads to the last piece of evidence, E9.

The example analysis path is an abstraction and not fully representative of the analysis process. While the analysis does tend to follow a general path of data correlation, the analysis also involves cross-validation of the data. Suppose a hacker broke into a Linux server and deleted all traces of his presence from the log files to deceive, or at least frustrate, an investigator. In this type of case, the investigator cannot rely on correlation alone; he must cross-validate evidence to confirm their integrity.

Cross-validating data can be performed in two ways, the first of which involves examining multiple pieces of evidence at a particular system or network level. To cross-validate log entries, for example, an investigator examines more than one correlated log file to discover if one of the log files was intentionally altered. Another example is the use of MD5 checksums to verify the integrity of a system binary. Both of these examples test the integrity of a piece of evidence against that of another piece of evidence or a known fact. Cross-validating does not guarantee the integrity of evidence, but it does provide a means for uncovering tampering at a single layer.

The second type of cross-validation is the analysis of a single piece of evidence across multiple layers. Every piece of evidence can be viewed in terms of multiple layers of abstraction. Network packets are well known for their layered structure, but files are also

stored as layers. For instance, a source code file can be viewed at the application layer by its affects when compiled. At the file system layer, the source code contains more attributes, such as read-write permissions, filename, and access times. Further down, the file is shown as a chain of linked file sectors, and at an even lower layer, each sector is a block of bytes. Cross-validating a single file is useful in cases where a hacker alters a single layer but fails to do so to the remaining ones. For example, a hacker may store data in the unused portions of file sectors in a GIF image file. To the file system and graphics applications, the file appears completely normal. However, this hidden information can be discovered by calculating the size of the data actually used by the file and the size that the file system reports that the image uses.

2.1.4 Post-Analysis

Mitigation and full incident patching are performed once the analysis phase is complete. Mitigation involves swiftly correcting the damage and cause of the incident, both of which are discovered during the analysis phase. This is the critical damage control phase. Full incident patching is performed after mitigation and is less time-critical than mitigation. The goal of full incident patching is to ensure that incidents similar to the one that occurred do not happen again.

Mitigation

This critical phase is dependent on the requirements of the system and the nature of the incident. The system is placed in a test environment or is not be visible to outside users. Appropriate steps are taken to return the system to normal via patching the vulnerabilities and damage relating to the incident, and if possible, via system backups. The system is next tested to ensure that critical functionalities have not been lost. Finally, the system is returned to its normal status and environment.

Full Incident Patching

Whereas mitigation stops immediate problems, full incident patching ensures that the system is safe and stable. This phase is the most thorough and time-consuming of the forensic steps. It requires further analysis into the cause and effects of the incident to make

certain that previous analysis is complete and correct. Every system component that relates to the incident should be tested to ensure that the incident has been corrected and that the mitigation did not cause other problems. Depending on the severity of the incident, the manufacturer should be contacted to report the incident and obtain additional patching advice.

2.2 Analysis Methods

The thesis has thus far presented the four major phases of the investigation and how they are related. This section examines the analysis phase in greater depth by showing the three major analysis techniques. Together, these techniques provide the logical tools used by investigators to uncover clues and correlate evidence.

2.2.1 Baseline Analysis

Baseline analysis is the juxtaposition of a known "good" state of the system against the current system state. It requires that a relevant baseline be created when the system is known to be operating correctly and is uncorrupted. This can be an image of the system or documentation of system attributes. Baseline analysis is a structured technique that allows the investigator to quickly determine which system attributes have been altered.

A typical baseline analysis using an image proceeds by first running an automated tool that checks for differences between a baseline and the current system. These differences could include the number of loaded modules, the available system resources, MD5 checksums of modules, and configuration files. The tool reports that numerous differences exist between the two system snapshots. The investigator then sifts through the differences to determine which are benign and which are not. From the list of malignant differences, the investigator is able to draw a conclusion about the nature of the incident and what mitigatory actions need to be taken.

Baseline analysis using documentation is different than image-based baseline analysis. The documentation baseline only covers the information that is considered to be useful for future investigations. The baseline cannot be complete; a complete baseline would entail too much information to be documented, thereby defeating the purpose of a documentation

baseline. This information needs to be concise for it to be useful to the investigator.

The advantages of baseline analysis using a system image are that it can be automated by a tool that compares the baseline to the current state of the system, the process of finding and patching problems is fast; and the process is minimally invasive. Automation is the key advantage, since the task of identifying problems in a system can be tedious and time-consuming. With baseline analysis, the possibility of overlooking problems or excessively long investigations are no longer threats. Baseline analysis is also useful in the case of a documentation baseline, since the documentation serves as a checklist of evidence to analyze and compare.

Baseline analysis also has several disadvantages. First, the requirement for a pre-existing baseline is rarely met. If a baseline does not already exist, a baseline from the factory media must be created to perform baseline analysis. This, however, can be time prohibitive. If a baseline is available, it needs to be relatively accurate for the comparison to the current system to be productive. A baseline of the initial system configuration or factory media provides some clues as to how a normal system appears, but the most useful information for an investigation is a listing of changes made to a system's configuration. It is the changes that allow the cause of the incident to be concealed amongst the various non-factory changes. However, system configurations rapidly change and few systems take proactive forensic measures such as baselining.

2.2.2 Common Vulnerability Analysis

Common vulnerability analysis works best when an incident appears to have been caused by a single, well-known type of event. This technique involves searching a database of common vulnerabilities related to the incident. The related vulnerabilities are then investigated by searching the collected information for corroborating evidence. Once a vulnerability has been matched to the incident and pertinent evidence, the investigator can proceed to patch the system accordingly.

This type of analysis has the advantage of being extremely quick in terms of identifying the cause of an incident and its effects. The database search is simply a matter of consulting the vendor's vulnerability database or a public one, such as CERT. When the advisory is found and verified, the analysis phase is complete, since the advisory provides the details of the incident and corrective measures.

One possible situation for common vulnerability analysis is that the vulnerability is not present. Searching a vulnerability database is not expensive, but finding a potential vulnerability in the database and trying to gather corroborating evidence can be. If all signs of an incident point to a denial-of-service attack, the first step of common vulnerability analysis is to search the vulnerability database for that type of attack. If two such vulnerabilities are found, the investigator must find the evidence that corroborates both stories. The investigator may come to the conclusion that neither attack occurred. However, the evidence collected for corroboration may provide clues to the investigator that assist with uncovering the actual cause.

The main drawback with this technique is that it does not provide a thorough investigation and is limited in its usefulness. In the case of finding the vulnerability in a database, the extent of the incident is probably not limited to the single vulnerability. Suppose an attacker exploited a known vulnerability to gain access. The attacker is probably attempting to make use of some resource and may establish a backdoor to maintain access to the resource. The vulnerability is only the means to an end - an end to which vulnerability access is impervious. Further investigation is thus required to ensure that other system damage is not present. Common vulnerability analysis does offer a strong method of uncovering the root cause of the incident, but does in itself not form a complete investigation.

2.2.3 Timeline Analysis

Timeline analysis is comparable to baseline analysis, but it offers several benefits in terms of forming a conclusion about the cause of the incident. An incident generally causes some anomalous system behavior resulting in the generation of system alerts. The investigator can use the knowledge of the incident time to generate a timeline of events that led to the incident. The technique is to analyze logs, scheduling information, and memory to develop a rough timeline of events.

Timeline analysis is an integral part of most investigations. Having a detailed timeline that contains the events preceding and following the incident gives the investigator a precise understanding of what caused the event and what conditions allowed for the incident; equally important are the events following the incident. They provide information regarding the extent and effects of the incident, which is crucial for mitigation.

This type of analysis is excellent for producing conclusive investigations, but it does have its problems. Timeline analysis requires that the majority of events around the time of the event be available and correct for an investigation to be effective. If a hacker is able to cause enough confusion regarding the time of certain incidents, then the timeline drawn from the incident is only an estimation of what happened at certain times. In addition, the timeline events must be consistent with the other events' time if different mechanisms record the event times; that is, the different mechanisms have unsynchronized clocks. This synchronization problem is especially important when considering a network-based attack where the logging mechanisms operate on multiple systems, e.g. firewalls, intrusion detection systems, and file servers. The investigator must be careful to ensure that the times are consistent and correct.

2.3 Current Analysis Methodologies

This thesis has thus far presented the general methodology for conducting an investigation utilizing the four phases, and it has also examined the main techniques used in forensic analysis. The methodologies for forensic analysis, however, have not been examined. The current forensics literature does not offer a satisfactory methodology for performing analysis. The literature only discusses general techniques and suggestions, or it offers methodologies that are unsatisfactory. The absence of a satisfactory, general methodology has resulted in *ad hoc* analysis. This section first evaluates the two general analysis methodologies that have been proposed, root cause analysis and semantic integrity checking analysis. Both of these methodologies offer some benefit, but they are both not entirely satisfactory. The third subsection presents the *ad hoc* analysis approach by examining several demonstrations from computer forensics literature. The remainder of the subsection examines the reasons behind *ad hoc* analysis and why *ad hoc* analysis is not the ideal methodology.

2.3.1 Root Cause Analysis

An incident is typically modeled as a series of events that began with a set of causes and resulted in a set of effects. Reasoning about an incident in this manner allows

an investigator to model the analysis as a timeline of events, with each event being correlated to other events. Peter Stephenson put forth an analysis methodology based on this idea to model an incident in terms of the cause of the incident [45]. The methodology is designed to pinpoint the exact set of causes that enabled an incident to occur so that appropriate countermeasures can be applied to prevent its reoccurrence. The justification for the work is that no formal computer forensics analysis methodology has been demonstrated, and computer forensics faces the unique problem of performing a valid investigation while coping with an extremely large set of data.

The modeling of the investigation is performed after the analysis is complete. That is, the root cause is not determined until a sufficient amount of evidence has been analyzed and the analysis is considered to be complete. First, a narrative of the steps taken during analysis is considered. The narrative is simply a listing of the steps taken by the investigator and outside, non-digital information that was made available during the investigation. Next, each step of the narrative is codified to a formal language that is a derivative of Lisp, called DIPL. Finally, the DIPL instructions are modeled in a Colored Petri Net (CPN).

This methodology offers several advantages. First, the methodology is truly formal, in that it requires a series of steps to be taken and makes use of existing formal techniques. Second, the CPN modeling of the investigative steps demonstrates the soundness of the analysis. If any steps were omitted or erroneous, the CPN would show them. For example, if an investigator makes the conclusion that an attack occurred at a specific time because of a single log entry, then that log entry must be correlated with other evidence, otherwise the CPN modeling will show the weakness of this claim. Third, the modeling can be viewed from different perspectives - such as jurisprudence, security policy, or technical - by checking a subset of the CPN model. Finally, the analysis is made available in a coherent form that can be quickly used to mitigate the incident effects.

While the methodology is certainly a step in the right direction, it does possess several striking disadvantage. First, the underlying logic of DIPL is incomplete with respect to the types of events that can be modeled. DIPL is designed to solve a specific set of cases, so its language is capable of supporting those cases and several other derivative cases. The problem with this language, like most security modeling languages, is that the set of environmental conditions is infinite. Any such modeling inherently possesses a level of indeterminacy with the actual events. As such, the investigator must adapt DIPL on a per-investigation basis. The *ad hoc* additions to DIPL are costly, and the resulting CPN may

not be correct if the investigator does not properly model the new event type. Therefore, this form of modeling is still susceptible to the "junk science" arguments that plague *ad hoc* approaches.

Two other disadvantages are that the methodology does not provide direction to the investigator during the investigation, and the CPN does not model incident effects. An analysis methodology should be able to direct the investigation and ensure that the analysis is sound with respect to the steps taken. Root cause analysis validates the analysis, but it does not ensure that the ongoing analysis is correct. Root cause analysis is valuable as a proof of the analysis, so it is compatible with future analysis methodologies that are applied during the analysis phase. The other disadvantage is that this language does not model effects. The methodology is of course designed to highlight the root cause of an incident, but most investigations are performed with effects in mind. The solution is most likely to extend DIPL to model effects, but this would drastically increase the complexity of the language. Extending DIPL would also lead to the same problems of indeterminacy of the language.

Root cause analysis is certainly viable for certain cases. The main application is for extremely complex cases in which an investigator must be able to pinpoint a cause when many possible causes exist. The methodology also successfully shows that a formal methodology is possible for computer forensic analysis. If the author is able to solve or mitigate the problems of indeterminacy with the use of heuristics, perhaps this methodology may be viable. Until then, however, this methodology possesses too many problems to be practically viable.

2.3.2 Semantic Integrity Checking Analysis

The most common technique for proving the validity of a piece of evident is to correlate and cross-check it with other evidence. Data redundancies exist in many locations, e.g. as file system layers, log entries, and operating system-supplied information. When an incident occurs, traces of that incident's occurrence are stored in multiple locations. Rarely is all of that evidence modified to some incorrect and consistent value. If a hacker breaks into a system, he must be able to mask his entry and presence. This requires him to alter various logs and system information to produce a consistent timeline of events that show that he did not break into the system. The attacker may modify the obvious information,

but other information most likely remains uncorrupted.

The notion of data consistency is considered by Tye Stallard and Karl Levitt in their proposed analysis methodology called semantic integrity checking analysis [44]. Since some data redundancies typically are retained by the system, the analysis can be formed using data invariants. An invariant in this context is a set of evidence that provide redundant information and that are not easily corrupted. For example, the application-level logs, network logs, and IDS entries can be combined to form an invariant about an intrusion. All data relating to that invariant is then searched for and examined. The juxtaposition of the invariant and the related data is evidence about the validity of one another. Furthermore, if inconsistencies exist, then the analysis can hone in on those data to look for signs of data corruption, which further develops the conclusions about the timeline of events.

Data invariants are useful for investigations that involve possible data corruption and a relatively high degree of complexity. Modern operating systems store a wealth of information in many different locations, so the availability of data invariants is high. The investigator applies his knowledge of the system and the nature of the incident to establish a small set of data invariants. From these invariants, specific information about the incident is examined and validated. This semantic integrity of the invariants also uncovers the inconsistencies in the cases were data has been corrupted. This methodology is therefore useful for minimizing the amount of data that needs to be examined while also quickly validating the correctness of the data.

The main problem is that this methodology is essentially glorified timeline analysis. The assumption is that a high degree of granularity and time consistency exist, which is often a poor assumption. The fine granularity refers to the amount of accuracy and degree of precision that time is recorded. For example, Microsoft's FAT file systems creation timestamp is accurate within ten milliseconds, written is accurate within two seconds, and accessed is accurate within one day, whereas their NTFS file system is the same for creation and written timestamps, but the accessed timestamp is accurate within one hour [1]. If an investigator is analyzing a system that incurs hundreds of thousands of file accesses per day, such as a web server, then low granularity requires him to account for all of the time in-between the time of the event and the different caused by the granularity. Time consistency is similar. Most modern computers are inter-networked. Information is being received from outside systems, so all of the timestamps from the outside computers must be considered. If another system has a system clock that is very inaccurate, then the resulting

investigation may be skewed if the discrepancy is not considered. These issues show that timeline invariants are inaccurate.

A host of other types of invariants are also examined, but they are also problematic. The well-known security tool called Tripwire uses invariants to periodically produce alerts about potential intrusions [48]. This proactive tool checks for file accesses, file checksums, and file sizes to determine if unwanted activity occurred on a system. Semantic checking integrity operates in the same manner, but it does not have the luxury of being proactive. Many organizations do not store information about system attributes and what defines a normal system state. Instead, the computer forensic investigations must be reactionary to incidents. The issue of checking the checksums of files is not a problem for operating system files or binaries; the investigator computes the MD5 checksum of those files and then compares each to known "good" copies of each. This type of invariant relationship does not work for user-space files and binaries. The checksums of these files in their "good" state do not exist. In the case that the system operates under a defined security policy or in the case that configuration backups exist, the configuration can be checked to uncover evidence. This tactic, however, does not guarantee the absence of deception by the suspect or the complete presence of all relevant events.

Semantic integrity checking is an attempt to provide a general methodology for conducting an investigation using known "good" evidence. This methodology is best suited for investigations that possess a small- to mid-range of complexity. In highly complex environments, the notion of invariants degrades into an almost meaningless form. For invariants to be successful, the system must be fine-grained with respect to the data, and the data must be consistent. In addition, a prior list of "good state" information is almost imperative when the system is fairly complex. If not, the methodology requires the investigator to use *ad hoc* techniques to complete the investigation.

2.3.3 Ad Hoc Analysis Approaches

The dearth of general, structured analysis methodologies exists in computer forensics literature. Practitioners and researchers alike have seemingly ignored this issue and have relied on a handful of analysis techniques and the expertise of investigators with respect to investigations. Most of the suggested techniques are rooted in practical considerations, whether they be investigative goal-related, system-specific, or pertaining to a particular type

of analysis. This section examines three examples of the current state of forensic analysis to demonstrate the general *ad hoc* approach that is employed.

Log Analysis

Perhaps no type of analysis is more representative of forensics than log analysis. A system can be set up to log virtually any information and system state; data typically collected include network packets, the status of scheduled processes' activity, intrusions detected by an intrusion detection system, process-specific alerts, and user logins. The difficulty is that the logging of these events occurs with a number of possible discrepancies between the log times, including different time granularity, the use of unsynchronized clocks, and the modification of various logs by an attacker. The investigator must be able to decipher these discrepancies to determine what information is correct and/or useful. This subset of forensics represents the greater problem of analyzing volatile and untrustworthy evidence. A hacker can corrupt logs; he can also deceive the investigator in other ways. He can replace system processes with trojan processes, hide information inside of steno-graphic files, and wipe the file system of any data residing in slack space. All of these anti-forensics techniques complicate the investigator's task of arriving at a complete and consistent conclusion about the causes and effects of an incident.

In a 2004 book on computer forensics, Cyrus Peikari and Anton Chuvakin wrote a piece on the topic of log analysis where they compared the field more to an art than a science. "Log analysis is not a science by a long shot, at least not currently; reliance on individual analysts [sic] skills and intuition as well as pure luck play too large a role in this endeavor for log analysis to qualify as a scientific pursuit" [34]. That is, the complex task of drawing conclusions about volatile pieces of evidence, any number of which may be incorrect - is too difficult to formalize; an investigator must use individual skills to effectively perform such analysis.

Peikari and Chuvakin do discuss the general techniques for log analysis. The main idea is that all possible logs should be correlated. The correlation will show the anomalies and discrepancies between the logs. The discrepancies may have always existed, but the discrepancies should be uniform and consistent. That is, normal system behavior would result in a consistent discrepancy. If a hacker tried to alter several logs to mask his presence on the system, the log analysis should show a discrepancy at that time. The hacker may, if

skilled, be able to uniformly modify all of the logs so that they are consistent. In this case, log analysis may not initially lead to any results. This is a case when log analysis is truly an art. A skilled investigator may realize that the logs are too normal, as it were, and may begin testing the sanity of the logs. For example, he may try logging into the account that is mentioned in the logs and examine the logs to see if the logged events are similar.

Despite only being a subset of forensics proper, Peikari and Chuvakin's discussion of log analysis and its lack of scientific rigor reveal it to be analogous to the current state of forensic analysis. The authors discuss the technique of correlation and how that technique alone does not direct an investigator's analysis. The use of a handful of analysis techniques, such as correlation, provides the investigator with a general principle by which he can verify data. The technique does not assist the investigator with the problem of organizing analysis and how to satisfactorily demonstrate when evidence has been sufficiently correlated.

NIJ Guideline Analysis

In July 2001, the U.S. Department of Justice's National Institute of Justice (NIJ) published a 93-page document aimed at providing forensic investigations with a set of guidelines for performing full computer forensic investigations called "Electronic Crime Scene Investigation: A Guide for First Responders." The document outlines the basic four-phase forensic investigation model is presented in this thesis: preparation, data collection, analysis, and reporting. It defines key issues, such as what constitutes a computer-based crime, a forensic investigation, and digital evidence.

The document also provides what the NIJ considers to be best practices for computer forensic investigators. These best practices range from the types of media that should be investigated, to how to properly collect data, to documenting the investigation, and to the transportation and storage of evidence. The NIJ wants to prevent the use of *ad hoc* practices that investigators performed in order for computer forensics to become a more standardized field.

The section on data analysis, however, does not provide a methodology, and it does not even offer investigators tips on handling complex investigations. Instead of presenting a basic methodology, the document outlines 14 of the most common computer crimes and the types of data that should be investigated for each crime [15]. For example, in the case of computer-based extortion, an investigator should examine emails, notes, letters, date- and

timestamps, Internet activity logs, temporary Internet files, and user names. If these types of evidence are fully examined, the document asserts, a sufficiently-strong case is formed.

This data type-based analysis methodology is not extremely useful on its own for complex cases. The NIJ document is targeted at law enforcement, as the authors state in the Foreward section, and because of this intended audience, the techniques and methodologies are focused on properly collecting and preserving data. The underlying belief is that a team of investigators will effectively analyze the case, but those investigators should be told how they handle said case. Law enforcement investigators are not concerned with expediting investigations. This lack of time requirements means that the analysis does not need to be expedited, which is the case during incident response. The authors thus do not provide investigators with a methodology to perform analysis quickly; the authors provide clues that aid the investigators in the data collection and analysis so that the investigators are less likely to botch the investigation. Suppose a single investigator bases the investigation of a computer intrusion on the analysis of the nine types of information proposed by the NIJ [15]. If the investigator wishes to expedite the investigation, he is thrilled that only nine different types of data need to be collected. After analyzing these data, such as temporary Internet files and address books, the investigator realizes that not only is this amount of data insufficient to solve the case, but that it reduces the problem of analysis on a large system to a small set of data - any of which may have been altered by the intruder. Frustrated, the investigator must now make sense of this data and organize it independently. In other words, he performs an *ad hoc* analysis.

Operating System-Specific Analysis

Instead of using a set of techniques to determine to investigate, an investigator may to follow a checklist of operating system-specific items to analyze. The investigator typically knows what operating system is involved in the incident. He can then follow a checklist of items to examine on that operating system. This checklist should ensure that a complete investigation is performed. The checklist prevents the investigator from prematurely concluding an investigation or forgetting to analyze potentially useful data.

Kruse II and Heiser provide checklists for performing analysis using varying levels of rigor for both Windows and Unix operating systems [24]. The checklists are not exhaustive lists that contain all data that could be useful to any type of investigation; the

checklists are subdivided into classes of incidents, e.g. application compromise or user account compromise. Kruse II and Heiser propose three degrees of rigor: minimal, serious, and fanaticism; the number of items from the checklist that are examined. Their justification for a checklist-based analysis is that the checklist provides non-savvy investigators the ability to perform a correct investigation. This is especially important because many system administrators have to perform forensic investigations, while the practical limitations of time, money, and training prevent many from becoming sufficiently skilled to be able to perform investigations without the use of a checklist. For the experienced forensics investigator, the checklist technique leads to reproducible investigations, since every investigation of a certain type follows the same checklist.

This technique has several disadvantages. First, few forensic investigations can be pigeonholed into a single type of compromise or incident that occurred on a single computer. Most company networks consist of multiple operating system platforms. Investigating each using the checklists may not provide sufficient correlation between the different operating systems. Moreover, the checklists most likely do not collect equal types and amounts of data for each operating system, so the data collected from one operating system may not be consistent with the data from another. Presenting a second difficulty is the fact that, even though systems contain proprietary or uncommon software that may require investigation, the investigator may fail to recognize this fact if he does not investigate outside of the confines of the checklist. The final disadvantage is that any analysis of the data must be performed in an *ad hoc* manner. The checklist alone does not tell the investigator how to correlate data or uncover signs of hacker deception.

The use of a checklist, whether it is specific to a particular type of operating system or type of incident, does not aid the investigator in performing a properly structured organization. The checklist acts as a crutch for the investigator - providing him with a cheat sheet for what data to collect and analyze. Once he has examined all of the data from the checklist, he must still arrive at the meaning of the data as a whole. That is, he must be able to make sense of the data by not only examining the data at a low level, but also abstracting the meaning of those low-level details into a high-level picture of what happened. The checklist is a purely *ad hoc* technique for aiding a hapless system administrator; it cannot be considered a proper analysis technique or methodology.

Reasons for Ad Hoc Analysis Approaches

Investigations occur under a wide range of conditions. Many of these conditions require that an investigation be carried out very quickly; the incident is to be analyzed and fixed as fast as possible. Investigations can also be less time-critical, e.g. an attack on a honeypot in a test environment. These two extreme cases each require their own analysis requirements. The former must limit the analysis of extraneous data and cross-validation, whereas the latter is afforded the privilege of analysis for completeness and cross-validation. The wide range of investigative conditions has led investigators to investigate in an incident-specific manner.

A second type of conditions is the type of system being investigated. While only a handful of operating systems are typically investigated, e.g. Windows, Linux, and Solaris, the configurations of each vary tremendously. The topology of a system's network and its configuration are critical for investigations, and corporations and government agencies are known to have their own in-house applications. The analysis must be structured to handle these. In addition, real-time systems, such as real-time databases and networking devices, have their own idiosyncrasies.

Investigators understand these incident-specific conditions and requirements and formulate their analysis plan on a per-incident basis. Their belief is that investigations are partly intuitive, which no methodology can capture. Since the incident-specific conditions do not lend themselves well to formalized methodologies, an *ad hoc* approach is used. This approach uses intuition and general understandings of investigations to find the way in the maze of evidence. The *ad hoc* approach is not a defeatist approach, rather it is considered to be a pragmatic approach to a complex problem.

Disadvantages of Ad Hoc Approach

The *ad hoc* approach is adaptable and intuitive enough to allow for any type of investigation; however, this approach is not without its faults. Inefficiency resulting from adapting an investigation for each incident is a problem. First of all, the investigator must not only spend time assessing the structure of the investigation, he must also be able to formulate an analysis strategy that minimizes the amount of extraneous data and dead ends. That is, the investigator needs to find an appropriate method for performing structured analysis that minimizes extraneous data. Inefficiency is indicative of an *ad hoc*

approach, and it is a non-trivial problem facing an investigator.

The second major problem with the *ad hoc* approach is its lack of rigor. The degree of precision and thoroughness of analysis is determined by the investigatory requirements. This leaves the matter of determining how to achieve said precision and thoroughness to the investigator. The issue of translating the investigatory requirements into a plan that provides sufficiently rigorous analysis is not completely determinate. The investigator must intuitively know when the investigation is complete, or if it is not complete, what information needs to be provided. An *ad hoc* approach provides no basis or directions for this problem.

The third problem is that analysis cannot be automated using the *ad hoc* approach. This problem stems from the undecidability of the *ad hoc* approach, since many of the decisions during analysis are intuitive and not purely rational. These nebulous decisions cannot be adequately mapped into a program that performs the bulk of the analysis. The implication of non-automatable analysis is that an investigator must always be trusted to perform an investigation and perform it well.

Chapter 3

Formalized Computer Forensic Analysis

Forensics analysis is perhaps the most critical phase of an investigation. While all of the phases overlap and are intertwined, the purposes of the other phases are to prepare for the analysis phase or apply its conclusions. The preparation phase provides a general understanding of the incident and leads to a preliminary investigatory plan. The data collection phase is performed such that all data needed for an investigation is collected and preserved. Finally, the post-analysis phase translates the analysis into a mitigatory plan.

Since the analysis phase is so critical, this phase deserves a rigorous methodology that is both efficient and thorough. Chapter 2 provides an overview of the phases and the problem with the current *ad hoc* analysis methodology. The current literature has not directly addressed a forensic analysis methodology. As such, investigators are left to perform *ad hoc* investigations that may be inefficient or incomplete. The absence of a true methodology has left a void in forensics that carries several consequences, including inefficiency, incompleteness, and the nonexistence of semi-automated analysis.

Several inherit difficulties with the analysis phase have prevented the development of a formalized methodology. First, the investigation process is dynamic in that its phases are not purely linear; the analysis phase may lead back to the preparation phase and back

again to the analysis phase. Since the phases are not linear, a methodology would have to account for the intertwined phases. Second, the requirements of every investigation differ in terms of time and completeness. These two requirements are often co-dependent, so the more emphasis placed on one negatively impacts the second requirement. A methodology must be adaptable to these requirements and the trade-offs between them.

This chapter contains the formalized forensic methodology that comprises the central contribution of this thesis. The methodology is designed to be simultaneously rigorous and flexible with respect to time and completeness requirements. The methodology's design draws from related data analysis fields: data mining, artificial intelligence, and intrusion detection. The logical techniques from those fields are applied to form the core analysis techniques for forensics. These techniques are interrelated units that are connected in such a way that they prove a general hypothesis about the incident.

3.1 Logical Approaches to Data Analysis in Related Fields

Three general steps occur within the analysis phase. The first step consists of collecting and organizing data in a meaningful form, and in the second step that data is scoured for anomalies and other forms of deception. In the third and final step, these data are then reorganized to form a convincing analysis result.

The similarity between the analysis of forensic data and the analysis of data in several other fields is striking. The first related field, data mining, searches across a set of data according to some known pattern, a type of analysis similar to the first two steps of forensic analysis. The second similar field is artificial intelligence (AI), namely the use of induction to simulate learning. AI learning analysis employs an inductive logic that formulates hypotheses and validates them, a technique which is comparable to analysis planning and reporting. The final related field is intrusion detection. Like data mining, intrusion detection searches for anomalous patterns and reports them; these correspond to the second and third steps of analysis.

This section provides an overview of the logic behind the analysis techniques of the three related fields. The logical techniques discussed in this section are those that are applied in the forensic analysis methodology. The discussion of each field's analysis is rather general, since the aim is to discuss the logical techniques used for analysis. The discussion

is not intended to examine the analysis methodologies or every analysis technique.

3.1.1 Data Mining

Data mining is a collection of techniques and methodologies for analyzing large stores of data. The mining process is an iterative sampling of data and pattern matching. A typical scenario begins with a large database of information that is not well understood. For example, a company may have a database of customer information that they wish to analyze for certain trends. The database is scanned for a particular data pattern or statistical pattern. These patterns are then interpreted, and the database may be scanned again for further results. After enough data are collected, the patterns are abstracted into representative data models. Figure 1 depicts the data mining process.

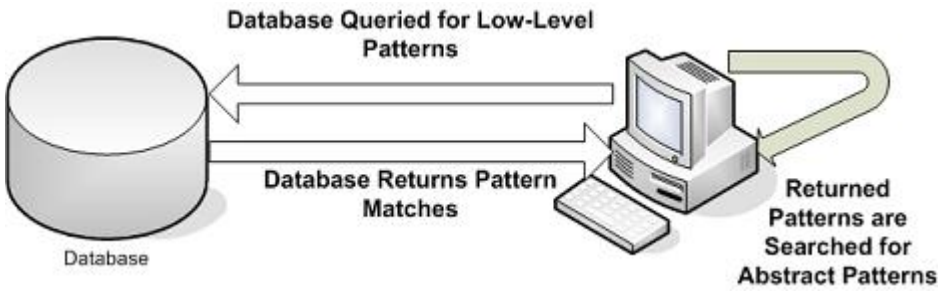

Figure 3.1: A basic data mining pattern matching model.

Pattern Matching

Data mining employs two levels of pattern matching. The first level is the conventional, low-level database model of pattern matching. The set of data is searched over by a known low-level pattern. When the pattern is found, those data are marked for later analysis or reporting. This low-level pattern matching is deductive by nature. Since the patterns and data set are known, the pattern matching results are known to be the set of all data from the original data set that match the pattern. Crosswise, all data outside of the result set are exactly the set of data that do not match the pattern. Databases perform

low-level pattern matching when employing a querying language - such as SQL - to locate data. The querying involves the examination of tuples for specific patterns, such as all tuples whose keys' values are greater than some integer constant.

The second type of pattern matching is a high-level, abstraction-based pattern matching analysis. After adequate low-level pattern matching is performed, the data are analyzed for various trends. The trends may be automatable, although not all data abstractions are fully automatable [52]. Abstractions are typically performed using well-known data models that are customized according to the nature of the data. The abstractions correlate data across data domains to form relations that did not previously exist between the data. The data domains[1] are searched for specific patterns that are used to derive conceptual information about the data.

An example of high-level pattern matching is an airline marketing group's customer profiling. The marketers may wish to build a model that represents the types of passengers that earn the most frequent flyer miles. The frequent flyer model would represent, among other facts, the most common age groups, regional locations, and flight types. This model is formed by taking the passengers with the most frequent flyer miles and tabulating their biographical information, which is then correlated with other frequent fliers' information. This frequent flyer model may not be sufficient for the company to begin marketing to a specific group, so the previous model can be abstracted further. The model would correlate regional locations and flight types to form a model that shows which flights in certain cities are most likely to contain frequent fliers.

3.1.2 Intrusion Detection

Intrusion detection aims to create systems and software that actively observe a system for possible intrusions via network connections or internal system methods, and there are two major approaches for creating intrusion detection systems (IDS): misuse detection and anomaly detection [4]. Misuse detection examines data for patterns that explicitly match known misuse patterns, whereas anomaly detection examines data for patterns that deviate from known acceptable use patterns. Since misuse detection is similar to data mining pattern matching [38], it is not considered in this thesis. Anomaly detection,

[1] Data domains are abstract groupings of like data. They may include similar entities or entities that are close to one another in terms of data proximity.

however, provides a different logical analysis technique.

Anomaly Detection

Anomaly detection combines system input and the system's acceptable behavior policy to determine if the input is an intrusion attempt. The acceptable behavior policy defines what states the system's acceptable states and what inputs are allowable. The policy's metric varies; software states, system call patterns, and input patterns are all valid metrics. The IDS monitors the system according the metric, so if the policy is defined in terms of acceptable system call patterns, the IDS monitors system calls. The input is then searched for in the acceptable behavior policy; the IDS signals an alert that a possible intrusion has been detected if the input is not found in the policy.

3.1.3 Artificial Intelligence

The AI analysis technique that is valuable for forensic analysis is learning[2]. The goal of all AI research is to create AI agents that can learn as well as or better than humans. This learning consists of applying predefined rules for gathering and analyzing data, which then lead to hypothesis formation and validation. This inductive process is comparable to human learning. An example of this learning is Deep Blue, the chess playing supercomputer created by IBM to challenge Garry Kasparov. In addition to the basic rules of chess, Deep Blue was created with a large set of principles, such as which moves to play at the opening of a game and how to evaluate a position. The most interesting feature of Deep Blue, however, is that IBM trained it by having it play sparring matches against another chess grandmaster [23]. In this manner, Deep Blue was taught how to play the objectively strongest move in numerous positions.

Hypothesis Formation

AI learning is an inductive process by which observed patterns are translated into new information. The first step in AI learning is the formation of a basic hypothesis, which is non-arbitrarily achieved by taking a priori knowledge and making a prediction about the

[2]This thesis' definition of learning is the formation of new data structures and new concepts from the analysis of a data set. This definition does not include the issue of intentionality.

unknown information. Next, the AI agent searches the data set for relevant data. This step is precisely the pattern matching analysis performed in data mining [5]. The information from the data mining step is then organized to form an inductive proof which either confirms or refutes the hypothesis. The AI agent then stores the knowledge of the hypothesis' proof or refutation for future use.

A hypothesis is initially formed by analyzing data for general patterns. The AI agent discovers a pattern and can begin testing for that pattern. After an adequate number of examples are found, the agent can assimilate that hypothesis into its memory. This type of hypothesis forming, however, can produce overgeneralizations of the hypothesis [29]. The agent must then search for counterexamples to the hypothesis so that the scope of the hypothesis is correct and not overly broad.

3.2 Formal Analysis Techniques

The forensic analysis methodology in this thesis is composed of several techniques that serve as the building blocks for the overall methodology. The most important of these techniques is the macrocosm hypothesis, which is the high-level proof that encompasses the entire analysis. Each of the macrocosm premises is composed of one or more lower-level proofs. Two types of pattern matching techniques are employed: high-level and low-level. The fourth type of proof, microcosm hypothesis, acts like a macrocosm hypothesis, only at a lower level. Together, these techniques form the foundation of the methodology.

This section explains the four analysis techniques. The techniques are discussed from the methodology standpoint. That is, the basic workings of the techniques are discussed, but the implementation details have been omitted. Each subsection defines the technique and discusses how the technique is applied and proven.

3.2.1 Macrocosm Hypothesis

The macrocosm hypothesis is the highest level and most abstract analysis proof of this methodology. It is the central proof for analysis; all other proofs all performed to facilitate the macrocosm hypothesis. This proof encapsulates all analysis, and it shows the overall result of the analysis phase. Because it encapsulates so much information, the

data are highly conceptualized and abstract, for abstracting is the only way that all of the information can fit into the proof without being convoluted by the sheer vastness of the data.

The macrocosm hypothesis is an inductive[3] argument whose premises are the conclusions drawn from lower level proofs. A hypothesis about the overall cause(s) of an incident is first derived. A set of basic conditions for the affirmation of the hypothesis are then produced. These conditions need not be complete or fully correct; they must simply, to the investigator's best judgment, provide initial criteria by which the investigation is to proceed. The first condition is then investigated via low-level analysis. If the first condition is proven by a low-level proof, then it serves as the first premise of the macrocosm hypothesis proof. Each of the remaining conditions is then proven, and if all of the conditions are proven, then the hypothesis is proven. The proving of the macrocosm hypothesis and the cases in which the conditions are incomplete or incorrect are considered in Section 3.3.1 and Section 3.3.2, respectively.

Formation of a Macrocosm Hypothesis

The central element of this analysis is the hypothesis itself. The hypothesis is derived from the incident report and the system specification, though it may also be derived from previous analysis if the hypothesis is reformulated. The hypothesis serves as a guide for the investigation, since all of the investigated low-level evidence must be tied to the hypothesis' proof or refutation. Since the hypothesis is the final analysis report, it may contain more than one incident cause. More causes in the hypothesis, however, detract from its simplicity, so fewer causes should be strived for and low-level causes should be contain within the hypothesis' premises.

The macrocosm hypothesis is derived based on the system specifications and the incident report. The system specifications provide the investigator with information about configurations, network connections, and general vulnerabilities. The incident report provides the investigator with specific problems that can be traced back to their original cause. These elements provide enough information to make an initial assessment about the nature

[3]This thesis uses the empirical definition of induction, which holds that a hypothesis is proven if the premises logically support the conclusion with only a small probability of error [17]. Empirical induction is akin to the scientific method and is not linked with mathematical induction.

of the incident,[4] and this approach to hypothesis formation is that of the AI hypothesis formation approach.

A system specification is a complete list of information about the system being analyzed. The two major components of the system specification are the operating system specifications and network specifications. The operating system specification lists the operating system type - including version and patch level - and the purpose of the system, e.g. an end-user workstation or a file server. The network specifications list all network connections and open ports, as well as the topology of the network.

The initial incident is a crucial set of data and concept that provides clues about the causes and effects of the incident. The information included in the initial incident includes the event and its time that first signaled that an incident occurred. An incident is signaled by a single event, but that event often elicits further investigation on the part of the person who noticed the anomalous event. Any further information that that person found, including the time(s), are to be included in this initial event record.

Tracking the time in which events are noticed and when they initially occurred is critical for creating a timeline. Timelines provide an investigator with focal points in time, which allows him to narrow the amount of evidence that needs investigated first. The timeline should include both the time of the events and the time that the events were noticed so that variations between those two times can be tracked. Together, the time of the events and the time that the events were noticed give the investigator a clearer picture of what type of events occurred.

3.2.2 Microcosm Hypothesis

The microcosm hypothesis functions in a manner similar to the macrocosm hypothesis; in both, a hypothesis is formed from known data, and lower-level proofs are constructed to prove that hypothesis. The microcosm hypothesis, however, operates at lower level than the macrocosm hypothesis. A hypothesis is formed and various low-level data are collected to form premises that prove the hypothesis. The premises are low-level evidence that strongly supports the hypothesis. This hypothesis is more technical and low-level than that of the macrocosm hypothesis; however, the microcosm hypothesis is still more abstract than that of its premises or a purely technical analysis. The reason for the mid-level detail

[4] See Chapter 4 for a case study demonstrate macrocosm hypothesis formation.

of the microcosm hypothesis is that it must be the intermediate form of evidence between low-level details and the macrocosm hypothesis.

Since the structure of the microcosm hypothesis is an inductive argument, this type of low-level analysis is best suited for situations in which data are missing or are contradictory to previously gathered evidence. Many investigations face the problem of missing data, as volatile data is lost or evidence is deleted. More troublesome than missing data is corrupted or deceptive data; this type of data has been intentionally modified to deceive the investigator by creating false events or a false timeline of events. The microcosm hypothesis is useful in such situations because it proves the cause of an incident with only a small probability of error. That is, while the microcosm hypothesis is not guaranteed, but the strong supporting evidence provides the investigator with the most likely event.

The microcosm hypothesis is intrinsically tied to the macrocosm hypothesis, but it is still not a truly low-level argument. The goal of the microcosm hypothesis is to prove the validity of a premise for the macrocosm hypothesis. The microcosm hypothesis must thus translate that premise into a series of low-level evidence that satisfactorily prove the premise. These premises are generated by taking the domain under which the premise falls - such as events relating to an IDS alert or all network connection attempts to a certain port - and combining with that the knowledge of the macrocosm hypothesis proof and system specifications.

Microcosm Hypothesis Formation

The macrocosm hypothesis determines the structure of the microcosm hypothesis. A typical investigation begins with an initial examination of some data, which are the initial clues. These clues then lead the investigator to some other clues that may have been unforeseen. This path-like analysis method is not completely avoidable, but the macrocosm hypothesis does provide the general structure of what high-level premises need to be proven for such an event to occur. The microcosm hypothesis is used to for the initial analysis of some low-level data. So the microcosm hypothesis is formed from the initial clues and system specifications that formed the macrocosm hypothesis, only the microcosm hypothesis is the follow-up on the clues. That is, the microcosm hypothesis proves the initial clues. If the analysis is beyond the initial clues, the microcosm hypothesis is formed by the next clue that needs to be proven for the macrocosm hypothesis. The formation of the microcosm

hypothesis also involves the AI hypothesis formation by gathering known data and making a judgment about what is probably true.

Proving the Microcosm Hypothesis

The proof for the microcosm hypothesis is defined by the requirements of the investigation. The base requirement for the proof is that a sufficient number of quality premises are supplied to verify the hypothesis. A single premise may be sufficient for a trivial or self-evident hypothesis. However, the requirements of the investigation determine the rigor and structure of the proofs. While a single premise may be sufficient for an unimportant or time-critical investigation, an investigation that requires great precision and detail may need multiple premises for the same proof. The meaning of proof, therefore, is determined by the degree of rigor that is required.

Low-level data is trivial to collect, but that information makes little sense in the greater context of an investigation. The data must be abstracted to a more conceptual form that easily translates into a single premise, i.e. the macrocosm hypothesis. The investigator must thus bridge the gap between low-level data and the high-level evidence by abstracting low-level data to an intermediate level. This intermediate-level evidence must properly represent the meaning of the low-level data but should strip away the technical details that weigh down the low-level data.

3.2.3 High-Level Pattern Matching

High-level pattern matching translates low-level data into intermediate-level evidence using pattern analysis. This technique is borrowed from data mining's abstraction-based pattern matching. Low-level data is first collected, and then those data are searched over using a suspected pattern. The pattern is either suspected based on previous analysis or it is an effective pattern for confirming or denying the existence of some anomalous pattern. The latter form of patterns is similar to that of anomaly detection: normal system behavior is assumed and anomalous patterns are shown by searching for the set of normal patterns.

Performing High-Level Pattern Matching

The process of high-level pattern matching involves collecting low-level data, scanning those data for trends, and then interpreting those trends. Low-level data are collected either in raw form or using low-level pattern matching. Once those data are collected, they are scanned for a particular trend. The scanning is typically performed using a semi-automated tool that allows the investigator to enter the pattern for which to search. The pattern may either be a suspected pattern based on previous analysis, or it may be a common pattern for a particular type of incident. In either case, the data that match the pattern are searched for. The investigator may search manually without a semi-automated tool, but this type of analysis is arduous, cumbersome, and error-prone.

Advantages

High-level pattern matching offers several benefits. This technique bridges the gap between the low-level data and high-level evidence, making the translation from low-level to high-level simpler for the investigator and less prone to errors. Directly translating low-level data to high-level evidence can lead to over-generalizing the implications of the low-level data, i.e. failing to consider the counterexamples to the analyzed low-level data, and directly translating can also lead to misinterpreting the low-level data findings. A second benefit is that traditional data mining techniques may be employed to quickly extract meaningful trends and patterns within the low-level data. These trends include the time of events, anomalous bandwidth usage, and super-user account usage. Another benefit is that this type of analysis is semi-automatable, so an investigator may use a forensics-specific data mining tool to search for a set of known patterns, thereby decreasing the amount of time spent doing such analysis.

3.2.4 Low-Level Pattern Matching

Low-level pattern matching is the simplest analysis technique, and it is the same technique used for low-level data mining. Low-level data - such as log files, recovered emails, and network packets - are searched using a pattern. The pattern is known and provides some information pertaining to the incident, such as event times, ASCII strings, or port numbers. The returned data is the set of all data that matches the pattern, which is, in

effect, a deductive proof of the data that match the pattern.

This type of data mining is performed very often without the investigator realizing that is being performed. Low-level pattern matching is the most natural and common analysis technique. For example, whenever log records for a specific time are being checked, low-level pattern matching is being performed. This analysis is the basis for all investigations, since data must be combed through at a low level using some form of data paring to eliminate extraneous data.

Cross-Validation of File System Layers

The following section illustrates low-level pattern matching using the example of the FAT file system. The FAT file system is comprised of seven layers; these are represented in the following diagram:

Layer	Input	Output
1	Raw file system image	Boot Sector values
2	File system image and Boot Sector values	FAT and Data Areas
3	FAT Area, FAT Entry size	FAT Entries
4	Data Area and Cluster size	Clusters
5	Raw cluster content and content type	Formatted cluster content
6	Starting cluster and FAT Entries	Linked list of clusters
7	List of clusters, clusters, formatted cluster content and type	All Directory Entries in a directory or raw content of a file

Figure 3.2: The FAT12 file system layers.

Each layer represents a different level of abstractness. The first layer, for example, is simply the entire file system image with no abstract data types; the third layer contains the file system's meta-data structures that describe the location and attributes of each file; the seventh layer are the linked list of clusters that form the directories and files along with their type. A forensic investigation may involve the low-level analysis of these layers for

potential clues of anti-forensics. For example, a hacker may alter the starting cluster value to point to an incorrect location, thereby hiding data.

One technique for doing low-level pattern matching for such FAT alterations is the cross-validation of the various file system layers.[5] The cross-validation consists of taking redundant data that exists in multiple layers and comparing these layers to ensure that no such corruption exists. If the layers do not match, then the file system has been corrupted at least one of those layers. For example, if the starting cluster value from layer three points to a location that is marked as unallocated (layer 6), then their comparison shows that corruption most likely occurred.

This technique is useful for low-level pattern matching for this thesis' framework because of the way in which a high-level hypothesis can be confirmed or refuted by such low-level evidence. Suppose that the current hypothesis is that a perpetrator attempted to hide data on a floppy disk. With the file system cross-validation, this hypothesis can be confirmed in the case that layers do not match, and data is shown to have been hidden by altering file system layers. Also, the cross-validation adds evidence against the hypothesis if no such irregularities are found, i.e. all layers are consistent. In both the former and latter cases, other techniques would need to be employed, especially at the application layers. Nevertheless, the cross-validation provides strong evidence at the file system layer.

3.3 Formal Analysis Methodology

The problems associated with the *ad hoc* analysis methodology discussed in chapter 2 show that an unstructured analysis approach can lead to inefficient and incomplete investigations. Using the *ad hoc* approach, an investigator must independently make decisions about the direction of an investigation, from beginning evidence analysis, to determining what constitutes a complete investigation, and how to reformulate his investigatory plan. If an investigator improperly manages the directional aspects of an investigation, inefficiency or incompleteness results.

The formalized analysis methodology presented in this section provides a structured approach by which an investigator can perform a complete investigation in an efficient and rigorous manner. Using the techniques presented in section 3.2, this methodology struc-

[5] The remaining discussion is from "Cross-Validation of File System Layers for Computer Forensics," from DFRWS 2003 [42].

tures investigations as proofs. The macrocosm hypothesis, the main proof, is composed in an axiomatic manner by the use of lower level proofs as premises. These lower level proofs may themselves be composed from other lower level proofs. The analysis thus contains low-level details at its foundations, while the high-level proof provides an abstract representation of those low-level details, making the analysis more understandable for the investigator.

Figure 2 is a graphical representation of an example analysis that applies this methodology:

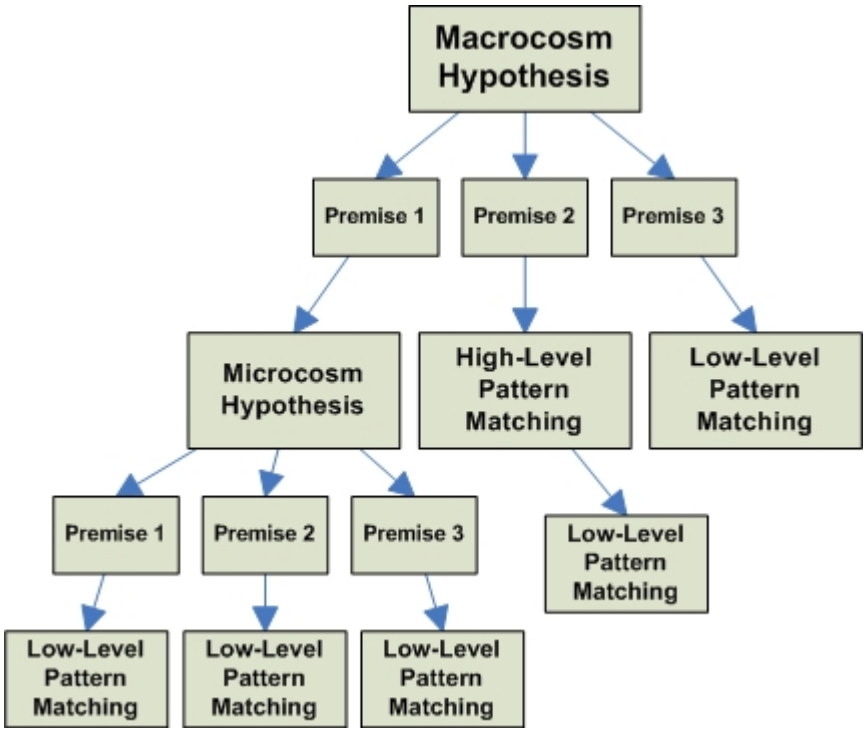

Figure 3.3: An example investigation flow.

This diagram illustrates an example investigation in which the macrocosm hypothesis is proven by three premises. The first premise is a microcosm hypothesis that is in turn proven by two low-level pattern matches. The second premise is a high-level pattern match that is derived from a low-level pattern match. The third premise is simply a low-level pattern match.

3.3.1 Initial Analysis

The traditional *ad hoc* analysis approach begins an investigation by examining a single clue to uncover further clues, which are in turn examined. The formalized methodology seemingly implies that the structure of the macrocosm hypothesis' proof must be known from the beginning of the investigation, since a scientific hypothesis is traditionally postulated once the criteria for its proof are known. The situation is different for this type of hypothesis. While structure is required of the final proof, the investigator cannot a priori know the exact structure of the evidence for each proof. The macrocosm hypothesis is thus formed, and the requirements for its proof are broadly stated. The requirements for the macrocosm hypothesis proof are that the hypothesis be proven completely and that no evidence that runs counter to this claim exists.

The procedure for producing macrocosm proof requirements is illustrated by the following example. Suppose a compromised end-user workstation is suspected of having been hacked and used as an FTP server for warez[6] distribution. The microcosm hypothesis can be stated as: "The workstation has been compromised and the hacker used it as a warez FTP site." This claim encapsulates the investigatory goals of the analysis. For the hypothesis to be proven, the analysis should prove each of the hypothesis' claims. In this example, the investigator must first establish that the workstation was indeed compromised. Then it should be proven, warez was placed on this machine by the hacker, and the third is that the machine serves the warez via FTP. The proofs of these three claims shows that a hacker performed the illegal activity and that the end-user was not responsible for the FTP server or warez. Had the end-user been responsible for either, the macrocosm hypothesis would be refuted.

Once the macrocosm hypothesis and its proof requirements are established, the analysis begins by examining the first clue. The first clue is taken from the initial incident report and is typically rather general. Since the clue is usually rather general, it should be formulated as a microcosm hypothesis. The microcosm hypothesis is applied to translate the general, high-level clue into a set of more technical, low-level elements. By decomposing the initial clue into a set of low-level elements, an investigator is then able to uncover further evidence and better understand the direction of the analysis.

[6] "Warez" is a hacker term that refers to all types of illegal software, e.g. commercial software, copyrighted songs, and copyrighted movies.

3.3.2 Proving the Macrocosm Hypothesis

Proving the macrocosm hypothesis is essentially the complete analysis phase; this proving is a complex, multi-faceted process. Section 3.3.1 discusses how the analysis begins and how the macrocosm hypothesis proof requirements are first established. These two elements of the methodology serve as the initial steps. The initial proof requirements are helpful for providing direction to an investigation, but they probably do not constitute the final proof requirements. This is because as analysis proceeds, an investigator should develop a keen understanding of what occurred and what type of evidence truly proves the macrocosm hypothesis. That is, the true complexity of a case cannot be known until analysis has begun. As details are uncovered, the focus of the analysis continues to shift to different elements of the incident until the macrocosm hypothesis is proven.

Before considering what constitutes a macrocosm hypothesis proof, one should understand the basic elements of the proof and the proof discovery process. The macrocosm hypothesis is proven by using a set of lower-level proofs as premises to show that the hypothesis is shown to be true. The premises may be any number of the three lower-level techniques, which are microcosm hypothesis, high-level pattern matching, and low-level pattern matching. Although the proof could be entirely constructed from low-level pattern matches, the proof should mainly be constructed from the mid-level techniques: microcosm hypothesis and high-level pattern matching. These mid-level techniques provide a layer of abstraction from the low-level pattern matching. Without these mid-level techniques, the low-level pattern matches would be the only premises in the macrocosm hypothesis proof, resulting in an overly-large, convoluted proof that consisted of every low-level detail. The mid-level techniques modularize the low-level pattern matches into more meaningful and compact forms.

Microcosm Hypothesis Proofs as Propositions

Apart from the macrocosm hypothesis, the microcosm hypothesis is the highest level analysis technique, and because of its level of abstraction, the microcosm hypothesis is well-suited for macrocosm hypothesis premise usage. The microcosm hypothesis must be proven by lower-level data, and this requirement implies that the hypothesis itself is an abstraction of the lower-level data. Using a microcosm hypothesis to prove the macrocosm hypothesis, in turn, implies that the macrocosm hypothesis is at least two layers of abstrac-

tion away from the low-level details, enabling the macrocosm hypothesis and its proof to be more compact and coherent. The abstract nature of the microcosm hypothesis is also advantageous in that all of the low-level details are not needed to understand the investigation at a later time. Simply knowing that an event occurred and that a proof consisting of the low-level events proves the event's occurrence is sufficient for an investigator to be convinced. If the low-level details are needed, such as in the case of post-incident recovery, the microcosm hypothesis proof can be traced back to its low-level details.

An added benefit of the microcosm hypothesis is that new clues are discovered during the proving of the hypothesis. The process of proving the microcosm hypothesis involves examining multiple sources of low-level data. During this process, an investigator notices a wealth of data that has not been considered. Within these sets of new data are patterns of information that are anomalous and somehow linked to the macrocosm hypothesis. For instance, an investigator may examine router logs at a specific time for activity on port 22, but he may also notice that numerous packets were sent to port 6667 around the same time. Since no traffic should be allowed into port 6667 and this activity occurred around the time of another known event, the investigator may wish to investigate the port 6667 activity further. An investigator should note any such clues and pursue their investigation after the current macrocosm hypothesis premise proof is complete. Waiting to investigate related clues until the completion of the current clue allows the investigator to exhaustively find all of the related clues and maintain the structure of the investigation, rather than resorting to an *ad hoc* structure. Figure 3 illustrates the discovery of a new clue during the proof of a microcosm hypothesis and its introduction as the next clue to be pursued for the macrocosm hypothesis proof.

High-Level Pattern Matching as Propositions

High-level pattern matching provides less abstraction than a microcosm hypothesis, but it does provide a layer of abstraction that is useful for the macrocosm hypothesis proof. A high-level pattern match condenses a set of low-level data into a single, abstract representation of those data. In doing so, it offers a conceptualized form of the low-level data for use in the macrocosm hypothesis. The pattern match typically occurs over a set of tightly related data elements, whereas the microcosm hypothesis proof occurs over multiple sets of data that may not be as tightly related. A high-level pattern is thus an abstraction

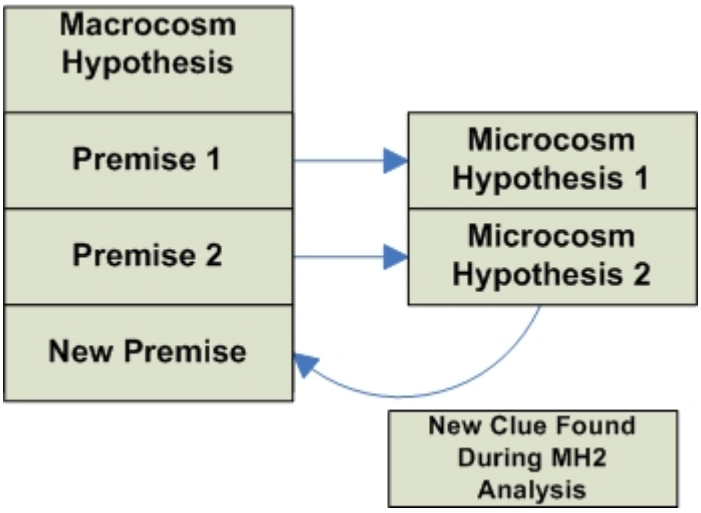

Figure 3.4: Discovery of a new clue during the microcosm hypothesis proof.

of a single set of related data.

High-level pattern matching is applied when the macrocosm hypothesis proof points to a clue that can easily be proven by generalizing a trend or other high-level fact about a set of low-level data. An investigator may have evidence of a clue that points to a set of related data. He may then use high-level pattern matching to verify this clue by extracting this high-level clue from the low-level data. For example, an investigator may suspect that the activity of port 21 increased greatly over a known period of time. He may then examine the router logs for usage of port 21, paying particular attention to the period of time in question. The high-level pattern is the activity of port 21, and this pattern either confirms or refutes the belief that the abnormally high activity on port 21 occurred at the specified time.

Validity of the Conclusion

The macrocosm hypothesis proof is the high-level representation about all of the analysis conclusions, but one must understand what constitutes a proof for the analysis to be correct. Speaking about proving the causes and effects of an incident seems straightforward and intuitive, but the strength of a proof depends on the complexity of the incident, which can lead to proof that may be debatable. For instance, an investigator may have to

construct a proof for a large company's web server, where even a single day of traffic exceeds 4TB [31]. The completeness of such a proof is surely questionable.

"Proof" denotes and connotes different degrees of certainty and believability, depending on the context [19]. The everyday form of proof that is conversationally used denotes a high level of evidence to which a rational person would agree is true. For example, a child can prove that he cleaned his room by showing his parents that his room is clean. The objectiveness of this proof is, of course, disputable. The child may have paid someone to clean his room, or he hid his toys and dirty clothes under his bed. The logico-mathematical notion of proof denotes the consistency of the premises and the conclusion, i.e. the premises necessarily imply the conclusion. This rigid notion of proof is viewed as the highest form of proof, since logical certainty is the perceived goal of all knowledge [18]. However, even logico-mathematical proofs are not completely certain. Euclid's axiomatic proofs for his geometry, the prime examples of mathematical proof, are not complete, due to the use of premises that are not contained within his axiomatic system [3]. Furthermore, groups of mathematicians are recognizing that abstract proofs are so complex that even they cannot always confirm the correctness of proof [22].

The macrocosm hypothesis proof is an inductive proof constructed according to the scientific method, and it is neither a conversational proof nor a logico-mathematical proof. The scientific method is the process by which a phenomenon is repeatedly observed, a hypothesis about the phenomenon is constructed, and the hypothesis is validated through repeated testing. This process is iterative, so a hypothesis can be refined or reconstructed if it is inaccurate or false, respectively. This iterative process is the same by which the macrocosm hypothesis is proven. Initial evidence is considered, and a hypothesis is formed. The hypothesis is tested against a series of evidence. These tests most likely provide the investigator with new insight into the incident, and the hypothesis is modified accordingly to reflect those insights. This iterative process is continued until the hypothesis is proven.

The point of contention in the scientific method is validation, since the degree of repetition needed for validation is subjective. The same applies to the macrocosm hypothesis proof; an investigator must be able to determine when a sufficient amount of evidence has been analyzed for the proof to be valid. Two factors help an investigator to decide when analysis is complete: experience and the investigation's requirements. An investigator's experience is important throughout an investigation; knowing the low-level workings of the operating system involved, how the forensic tools work, or how similar investigations were

solved are all going to impact the quality of an investigation, regardless of the analysis methodology. Similarly, analysis is also determined by the requirements of an investigation. If the investigation needs to be quickly completed, then the analysis methodology can only provide benefits within those time limits. When combined, these factors determine when the analysis proof is complete. The issue of knowing when an investigation is complete also applies to the *ad hoc* approach, since the investigator must know when enough evidence has been analyzed. With this thesis' methodology, however, the investigator has the analyzed high-level evidence to provide a more meaningful representation of the analysis progress.

3.3.3 Altering a Hypothesis

The initial macrocosm hypothesis is at best an estimate of the causes and effects of an incident. However, an investigator is likely to uncover evidence that complicates or completely changes the direction of analysis, so an investigator must be able to adapt to the changing conditions while maintaining the proof structure to the analysis. This altering of the hypothesis is to be expected, and it is a natural aspect of analysis.

Reasons for Alteration

The cause of every macrocosm alteration is that one series of low-level analysis disproves one or more aspects of the hypothesis. The macrocosm hypothesis should be altered whenever the hypothesis is inaccurate or completely false. The former situation occurs when new evidence is discovered that shows the hypothesis to be incomplete or misdirected. An incomplete hypothesis occurs when the hypothesis does not include one or more of the major causes and/or effects. For instance, a hypothesis regarding a denial-of-service attack may be incomplete due to the denial-of-service being a distributed attack. In the case of a misdirected hypothesis, the evidence may suggest that the hypothesis is mostly correct, but one or more details need to be modified. An example of a misdirected hypothesis is that during a denial-of-service investigation, the hypothesis may point to a certain set of IP addresses as the origin of attack, but later analysis shows that those addresses were spoofed.

In more severe circumstances, the macrocosm hypothesis may need to be completely altered. This situation occurs when a lower level evidence analysis completely re-

futes the perceived causes and effects of an event. The analysis changes the hypothesis significantly enough to require the previous analysis to be reformulated into a new hypothesis. That is, the previous low-level analysis must be juxtaposed against the new evidence to formulate a new hypothesis. For example, an investigator may believe that a user was the target of an email hoax, whereby the evidence from the email is considered with respect to the belief that the sender was the attacker. Upon further investigation, the investigator may come to the conclusion that the email was in fact a worm, so the hypothesis must be completely recreated.[7]

Reformulating the Hypothesis

The hypothesis is reformulated in the same manner as the initial hypothesis is formed. Making minor modifications to the hypothesis is trivial: the missing or incorrect evidence is added, and any extraneous or incorrect evidence is removed. Completely reformulating the hypothesis, however, requires the investigator to reconsider examined evidence and the investigatory requirements. The reformulation is achieved by reconciling the initial hypothesis with the new evidence. The new evidence typically provides additional clues, which may be incorporated into the hypothesis.

3.4 Caveats

While the process of formulating investigatory analysis as a formal proof seems intuitive, real-world analysis presents difficulties that prevent the methodology from being truly formal and applicable for all investigations. Forensics is a term that encompasses a wide range of investigations. These investigations range from a simple investigation of a worm, to a full-fledge federal investigation of a large number of hacked corporate networks. Because of this array of investigations, every investigation must be conducted in a unique manner. The methodology put forth in this thesis may be used in every investigation, but practical considerations make this methodology unreasonable in some cases. This section discusses the caveats of this methodology, namely the types of cases in which this methodology should not be applied, limitations on the hypothesis length, and required investigative skills.

[7] See Section 4.3 for this investigation.

3.4.1 Inappropriate Situations for Formal Methodology

This methodology is designed to provide the investigator with an organization for analysis in the face of complexity. Without the complexity, the methodology produces extra analysis overhead. There are two types of complexity that are required for this methodology to be useful. The first is that the investigation involve an incident with a multitude of possible causes and effects. This case occurs when the investigator must decide which analysis path to pursue when confronted with several good choices. When the analysis path is relatively straight-forward, then the proofs do not provide additional structure to the analysis; they simply add extraneous representations of the information. Also, if the analysis path is relatively straight forward, much of the analysis can be carried out using automated tools that eliminate the need for human investigatory skills.

The second type of complexity required for this methodology is a large set of data. Much like the complexity of multiple causes, the methodology requires that a sufficiently large amount of data be present for it to be useful. If the data to be examined is a small set of network logs or a single floppy disk of information, then the need for proofs is limited. The data can be examined much more quickly without the use of proofs, so the proofs are never needed.

3.4.2 Hypothesis Length Limitations

A second caveat is that the hypothesis must be limited in its length in some manner. Suppose a complex incident is caused by a series of twenty of more events. The hypothesis cannot be represented as a list of these twenty or more events. Instead, the events must be abstracted into one or two causes. The low-level details of the abstract representation should be contained within the premises. Otherwise, the point of using proofs is lost, and the investigator should instead rely on current methods, whereby the investigator makes a long list of events and tries to solve the cause from that information.

3.4.3 Required Investigative Skills

The final caveat is that an investigator needs to be competent at analysis for this methodology to be useful. An argument against this formalized methodology is that a skilled investigator performs investigations just fine without the use of proofs. This is

true. If a highly-skilled investigator performs *ad hoc* analysis, he will most likely arrive at a better conclusion than an unskilled investigator who uses this methodology. This argument, however, misses the point of the methodology. The methodology is designed to provide an investigator with a structured means for conducting analysis, regardless of skill level. An unskilled investigator may find more value in this methodology, but a skilled investigator should also find benefit in this methodology. First, the analysis is represented abstractly throughout the investigation, so a forensic team can better communicate their analysis to one another. Second, the documentation aspect of analysis can be subdivided into more logical forms, which provides the investigator with an clear, logical representation of the analysis when the investigator becomes stuck and must reformulate his analysis plan.

Chapter 4

Case Studies

This chapter presents a series of case studies that apply this thesis' methodology to the forensic analysis phase of actual incidents. The first investigation involves a hacked end-user's desktop system, and this demonstrates how well the methodology works for a complex case that involves several initial clues but also a large amount of possible evidence. The second investigation is a suspicious email that appears to be a worm but is not discovered by anti-virus software. This case illustrates the adaptability of the methodology in the face of a complete change in the direction of analysis. The third case is a low-level investigation of a captured floppy image, and this shows how the methodology provides a convenient representation of the investigation while maintaining the low-level proofs of the analysis conclusions. In addition, this case is taken from a forensic competition that pits forensics professionals against one another, so this methodology's conclusions are compared to that of a leading forensics investigator's analysis methodology. Finally, a case is taken from forensics literature, where the supposed correct analysis is shown; the author's analysis is then dissected to demonstrate the strengths and weaknesses of this thesis' methodology.

These case studies substantiate the versatility of the methodology and its other relative strengths, but they also reveal this methodology's deficiencies and limitations. Forensics encompasses a wide range of investigations, including incident response and improper usage examinations. For a methodology to be effective, it needs to be applied in the proper context. The case studies in this chapter are supplied to show how effective the methodology is in various contexts. While an ideal methodology would be useful in every scenario, this

is not the case. The varying levels of complexity and case-specific requirements prohibit an all-encompassing methodology. The present methodology is, then, shown to be useful when a case is sufficiently complex; however, it is sometimes overkill and may complicate an otherwise simple case.

4.1 Hacked Windows XP System

The following case study is an investigation of a typical Windows user's machine that is compromised and used for malicious purposes. The purpose of this case study is to examine a nontrivial hacking incident without any foreknowledge of the causes. The system was designed to be hacked so that a compromised end-user computer could be analyzed. The machine was set up with a default Windows XP installation with all of the default services left running. The operating system was not patched, and the machine was connected with a broadband connection and placed in a DMZ, so it was no behind a firewall. Security experts claim that such a system can easily be hacked within ten minutes [51]. The system was monitored twice every day for anomalous system behavior, such as mysterious running processes or bandwidth usage. The system's logs were not protected via a central logging mechanism, nor was an intrusion detection system or packet capturing mechanism put in place. That is, the system was not designed as a honeypot. The system was left online with the user logged on during the entire experiment, and the system was monitored via Windows' remote desktop sessions.

4.1.1 Background

The computer used for this experiment is a 2.4GHz PC with a 60GB hard drive and 256MB of RAM. The operating system was the default installation of Windows XP Professional with no service packs or other patches installed. The amount of free hard drive space at the start of the case study was 54.91GB, which accounts for the space occupied by the operating system and file system formatting. The computer was attached to a low-end, 4-port wireless router/firewall combination over a broadband Internet connection. The computers inside of the LAN had their addresses NAted. File sharing and Remote Desktop were turned on, so the default network ports, NetBIOS ports (135-139), and the RDP port

(3389) were open. Only one account, "Ricardo," was set up, which on Windows is the Administrator account. The system was placed online at 9PM on March 1, 2004.

The system appeared to behave normally for the first four days, but at 10AM on March 5, the system was noticeably performing very badly. The operating system responded very slowly to user activity. Text typed at the command prompt did not appear immediately, and the system's hard drive light was continually lit, indicating a high level of usage. The system was performing so slowly that the system was most likely compromised. This was confirmed when the hard drive space was checked, which showed that only 35GB of hard drive space was available, a reduction of almost 20GB.

4.1.2 Investigation

The probability of an incident having occurred was almost certain. The case study now shifts to a full investigation. The goals of the investigation are to return the system to its original running state without formatting and reinstalling, determine the causes of the incident, and to uncover every effect of the incident. With these goals in mind, the investigation should place more emphasis on completeness rather than timeliness.

The analysis is performed using the system itself as the site of the investigation. That is, the victim system will not be imaged and examined on a separate machine; the analysis will be performed on the original machine. The only evidence that needs to be collected is the volatile data, including system and network logs, memory space, running process information, and network connections. These pieces of evidence may become corrupted during analysis, so they were collected to ensure their preservation. Once this information was collected, the system was disconnected from the network, shut down, and rebooted using a bootable Linux CD that provides the forensic analysis environment. The bootable CD is called F.I.R.E. [11]. It is a full Linux operating system with a suite of freely-available forensic analysis tools. Using this environment, the Windows file system may be mounted as read-only so that no data is altered or otherwise damaged.

Initial Analysis

The initial analysis phase consists of the formation of an initial analysis hypothesis by combining the requirements of the investigation with the information pertaining to the

incident. The incident has caused the system's performance to deteriorate to the point of inoperability, and the hard drive has become filled with 20GB of new data. The incident apparently resulted in a hacker gaining access to the system and using it as a data storage system of some sort. In addition, since the since is an unpatched Windows XP system that was broken into within four days, the attacker most likely used a ready-made exploit to attack a well-known system vulnerability. The initial hypothesis is thus stated: "An attacker exploited a well-known Windows vulnerability to gain access to the system, and he then established a backdoor by which he could use the system to store data."

The first premise of the analysis is to prove that an attacker exploited a well-known vulnerability to access the system. This premise is a microcosm hypothesis, stating that the attacker exploited a well-known vulnerability, but the exact vulnerability is not yet known. The first step in proving the microcosm hypothesis is to examine logs to find system errors or other anomalous system messages. The system was scanned for worms, trojans, and other well-known malicious code using f-prot anti-virus software with the latest virus definitions; no such malicious programs were found, though. Next, the Windows log files are examined. The first noticeable clue is that the oldest log entry from all three types of Windows logs - System, Security, and Application - is dated 8PM on March 4. Since Windows retains logs for seven days or a maximum of 512KB of log data, the logs were cleared by the hacker sometime before 8PM on March 4. Moreover, the deleted logs imply that the hacker did acquire Administrator access to the system, since only Administrators may delete log entries. The remaining logs provided no clues about the intrusion.

The missing log files provide a clue about the time of the compromise sometime before 8PM on March 4. The system is next examined for suspicious events that occurred around this time. It is reasoned that the hacker most likely covered his tracks by erasing evidence that specifically points to the attack. That is, he probably did not completely remove all possible pieces of evidence, so the evidence that was erased most likely describes the type of events that occurred during the attack. For example, if a hacker breaks in by exploiting an FTP server's vulnerability, he will definitely erase evidence pertaining to the FTP server, network data, and user logins. He may not, however, erase evidence pertaining to the file system or the other servers, since those were not involved in the attack. The hacker's selectively deleting certain types of logs allows the investigator to focus his investigation to those logs. The microcosm hypothesis proof now shifts to timeline analysis, using the day of March 4 as the initial timeline. The pieces of evidence to be considered

are the following:

- Allocated files' Modified/Accessed/Created (MAC) times,
- Deallocated files' MAC times,
- User logins, and
- Miscellaneous log files' entries.

MAC times analysis is performed by using Autopsy, a front-end for the freely-available suite of forensics tools called TASK. The tool automatically sorts all files by modified, access, or created time. It also recovers all deallocated files and performs the same MAC time sorting. In this case, 152 allocated files' modified, access, and/or creation times matched March 4, and these are narrowed to 32 by restricting the MAC times to 7PM to 8PM on March 4. None of these files provide immediate clues, although they may be useful later when confirming the exploit. Over 20GB of deleted files are recovered by recovering the deallocated files. Most of these files, however, have timestamps from March 5. Also, the deleted files do not contain the missing log files, so the timeline analysis does not provide information about the time prior to the incident.

The next two types of evidence, user logins and miscellaneous log entries, are considered next. Windows only stores user logins in its event logs, but those were erased, so that evidence is unavailable. Miscellaneous log entries, located in the Windows system directory, are examined, but they also provide no useful information.

The lack of clues gained from the timeline analysis requires that the deleted system logs be recovered. The logs are stored in the C:\WINDOWS\system32\config\ directory as the regular files *SysEvent.evt*, *SecEvent.evt*, and *AppEvent.evt*. Autopsy is used to recover these three files. Each file is examined for error messages. *Application Event* displays the following error:

```
Event Type: Error
Event Source: EventSystem
Event Category: (50)
Event ID: 4609
Date: 3/04/2004 Time: 3:21:35 PM
User: N/A
Computer: RICARDO
Description: The COM+ Event System detected a bad return code during
```

```
its internal processing. HRESULT was 800706BE from line 44 of
d:\nt\com\com1x\src\events\tier\eventsystemobj.cpp. }
```

The following is the only event message listed in *Security Event*:

```
Event Type: Success
Audit Event Source: Security
Event Category: Logon/Logoff
Event ID: 551
Date: 3/04/2004 Time: 3:21:35 PM
User: RICARDO\Ricardo
Computer: RICARDO
Description: User initiated logoff: User Name: Ricardo
Domain: MSHOME Logon ID: (0x0,0x174c3)
```

System Event provides the following two errors:

```
Event Type: Error
Event Source: Service Control Manager
Event Category: None
Event ID: 7031
Date: 3/04/2004 Time: 3:21:35 PM
User: N/A
Computer: RICARDO
Description: The Remote Procedure Call (RPC) service
terminated unexpectedly. It has done this 1 time(s). The following
corrective action will be taken in 60000 milliseconds: Reboot the
machine.
```

and

```
Event Type: Information
Event Source: USER32
Event Category: None
Event ID: 1074
Date: 3/04/2004 Time: 3:21:35 PM
User: NT AUTHORITY\SYSTEM
Computer: RICARDO
Description: The process winlogon.exe has initiated the restart of
RICARDO for the following reason: No title for this reason could be found
Minor Reason: 0xff Shutdown Type: reboot
Comment: Windows must now restart because the Remote Procedure Call
(RPC) service terminated unexpectedly.
```

These logs show that a series of related events occurred on March 4 at 3:21:35 PM. Because they occurred at the same time and fall within the time range of the suspected incident, they are examined more closely. A Google[1] search for "eventsystemobj.cpp" and "line 44" results in 55 web pages that discuss the exploit against the RPC-DCOM vulnerability, which affects various Windows distributions, including XP. Two write-ups of this vulnerability and its exploit are examined to match the attack signature with that of the victim system [20] and [36]. This exploit provides the attacker with remote administrator access to the system, and with that, the user can install backdoors to maintain control. The following signature attributes confirm that this exploit was used to gain access to the system:

- Windows XP Professional systems are vulnerable to this exploit;
- all four log entries match the exact signature of the attack;
- a system service (svchost.exe) runs with a known set of threads and dynamically loaded libraries, all of which match a running process on the victim system.

This low-level evidence sufficiently proves the microcosm hypothesis: that a well-known exploit was used to gain access to the victim system.

Remaining Analysis

The next premise of the microcosm hypothesis is that the attacker established a backdoor. The RPC DCOM exploit provides a backdoor via TCP port 4444 that is opened by the exploit. The list of open ports gathered during data collection shows TCP port 4444 is listening, but more interestingly, that TCP port 6644 is also listening. Neither of these ports was listening when the system went online. These ports have thus been opened by the attacker, but the purpose of port 6644 is still not known. The Windows registry is scanned for the string "6644". The string is found in the registry's HKLM hive under /SOFTWARE/Microsoft/Windows/CurrentVersion/Run, with the entry "WinTask32" whose value is

```
"C:\PROGRA~1\Internet
Explorer\PLUGINS\Vanquish\nc.exe -l -p 6644 -e
cmd.exe."
```

[1] http://www.google.com

The attacker placed a copy of Netcat on the user's system, which listens on port 6644 and launches a command prompt when the attacker logs into the system. Since Netcat is run as a system service, it is displayed as svchost.exe on the victim system and automatically runs when the system is rebooted. This enables the attacker to maintain access to the system even after the victim system is patched.

The next premise is that the attacker used the system to store and/or transfer roughly 20GB of data. The victim system's open ports show that NetBIOS ports 135-139, port 4444, and port 6644 are provide access points for files to be transferred. Because of the weakness of NetBIOS, the system's shared folders are examined. The only share available is called "Shared Folder," which is located at "C:\Documents and Settings\All Users\Documents." Directory listings on this and all of the subfolders show over 20GB of data located in the "Vanquish2\Uploads" directory. The directory contains eight copyrighted movies in MPEG format and twelve commercial software packages. Since these were never placed on the system, the attacker must have placed these files there.

The macrocosm hypothesis was proven, and the investigation may be concluded here. The analysis showed that the attacker used the well-known RPC DCOM exploit to gain access to the system. Once the attacker gained access, he placed a Netcat listener backdoor to maintain his access, even after the system is patched; this is shown by the Registry entry that launches the listener whenever the system is rebooted. Finally, the 20GB of new data is discovered in the shared directory, indicating that the hacker uploaded the files via Windows' file sharing. One question remains: if the attacker took such great efforts to delete logs and maintain access to the system, why did he place his files in directories with "vanquish" in the name?

The system is now rebooted into Windows to examine these files. The directory containing the uploaded files,

```
"C:\Documents and Settings\All Users\Documents\Vanquish2",
```

is not found. Also, the directory containing Netcat,

```
"C:\PROGRA~1\Internet
Explorer\PLUGINS\Vanquish,"
```

is not found. The registry is checked for the Netcat entry. It does exist, but its string value, which should list Netcat's path, is blank. To confirm that Netcat is running, a telnet connection to localhost port 6644 is established. This does produce a new command prompt.

Next, the hard drive information is queried, which does show that the 20GB are still being used, so the files still exist. Finally, a file search for any file or directory containing the string "vanquish" is performed; no results are returned, however.

Since the 20GB are still allocated and the Netcat listener is active, the system must contain a rootkit. Rootkits are far more common for Unix variants, but several do exist for Windows. A Google search for "vanquish" and "rootkit" returns several results that describe a rootkit named, aptly enough, Vanquish. The vanquish rootkit is a DLL-injection, API-hooking rootkit that hides all files, folders, and registry entries that contain the string "vanquish" and stores all of the system passwords [53]. This information explains why the registry entries and folders containing "vanquish" were not visible from within Windows, but were when in Linux.

Post-Analysis

The victim system was set up with the intent of being hacked and subsequently investigated; there was no intent of recovering from the attack by patching the system and removing any malicious information. If the system were to be returned to normal, several additional steps would become necessary. First, the rootkit should be removed from the system. This is achieved by removing its DLLs and API hooks in Linux by manually deleting each from the mounted system image. Second, a search for "vanquish" should be performed to uncover all traces of data left by the attacker; all of these data should be deleted. The system should then be rebooted, and the Netcat listener should be removed from the registry. Finally, a Windows update should be run to install the latest security patches. The machine can then be rebooted, and all traces of the attack will have been removed.

4.1.3 Observations

This investigation demonstrated the relative strengths of this thesis' methodology. The attacker took efforts to conceal both the intrusion and the hacker's subsequent presence. These factors made the investigation more complex, since the traditional means for analysis were insufficient. The complexity required that the analysis be more focused and properly directed at uncovering evidence. That is, each step of the analysis needed to be significant

and not a random shot in the dark. Using the formalized methodology, the analysis was conducted with precision, and it proceeded with logical steps that proved each premise without straying from the investigatory plan.

If an *ad hoc* approach were employed, then the analysis might have diverged from the small, modular proofs of this methodology. For example, when the allocated files were examined during the analysis of the intrusion, an extremely large number of files were found inside of the shared folders. This might have roused interest in an investigator, since the primary effect of the attack is that 20GB of hard drive space was made unavailable. The investigator could either note this oddity and continue to look for the cause of the intrusion, or he could begin analyzing that directory for evidence of the missing 20GB. This latter, tangential analysis, however, would cause the investigation to become convoluted and unstructured. In turn, the lack of structure could lead to an incomplete and inefficient investigation. With this thesis' methodology, the investigator operates with the knowledge that each step of the analysis should be directed to the solving of a single problem, so he cannot stray from the analysis plan.

In addition to the benefits of structure, this case study shows that the methodology scales well for complex investigations. Because the proof-based analysis subdivides clues into smaller modular units, the investigator is able to investigate a large problem while maintaining a narrow focus without sacrificing the overall completeness of the analysis. In this case study, for example, the macrocosm hypothesis began with a premise that claimed an attacker gained access to the system by using a well-known exploit. Uncovering the exact exploit that was used against Windows is a daunting task when one considers the many sources of evidence on the system and the numerous exploits that are available to hackers. This complex problem was subdivided into smaller sets of clues, such as log files, open ports, and other attack signatures, and each of these clues was further subdivided, such that the solving of each of these sub-clues became an issue of performing a series of low-level analyses that result in the solving of the clues. Scaling is perhaps the most important benefit of the methodology, since the tasks of subdividing clues and performing low-level analysis are intuitive. This scaling mitigates complexity and assists the investigator with performing a complete analysis with respect to the requirements of the investigation.

4.2 Email Worm

A suspicious email was received on March 13, 2004, that informed the user that his account was going to be revoked due to terms-of-service violations. The email header was forged to show that the email came from the account's administrator. The following is the email header followed by the body of the email message:[2]

```
Return-Path: <john@doe-smith.com> Received: from
u324.anysite.com(u324.anysite.com[10.20.1.4])
     by anysite.com(Cyrus v2.1.11) with LMTP; Sat, 13 Mar 2004 15:42:41
     -0500
X-Sieve: CMU Sieve 2.2 Received: from localhost (localhost
[127.0.0.1])
     by u324.anysite.com(8.12.10/8.12.10/N.20031118.07) with ESMTP id
     i2DKgeeB012702
     for <victim@anysite.com>; Sat, 13 Mar 2004 15:42:40 -0500 (EST)
Received: from u324.anysite.com([127.0.0.1])
     by localhost (u324.anysite.com[127.0.0.1]) (amavisd-new, port 10024)
     with ESMTP id 12639-02 for <victim@anysite.com>;
     Sat, 13 Mar 2004 15:42:40 -0500 (EST)
Received: from webhost2.net (bgp213.major.isp.com[10.30.2.4])
     by u324.anysite.com(8.12.10/8.12.10/N.20031118.07) with SMTP id
     i2DKgaYi012678
     for <victim@anysite.com>; Sat, 13 Mar 2004 15:42:37 -0500 (EST)
Date: Sat, 13 Mar 2004 15:42:00 -0500 To: victim@anysite.com
Subject: Notify about using the e-mail account. From:
support@anysite.com Message-ID: <edwlaflmdrqcyirytdr@anysite.com>
MIME-Version: 1.0 Content-Type: multipart/mixed;
boundary="--------ihxbljdshsogbntcaobe" X-Spam-Status: No,
hits=4.2 tagged_above=-100.0 required=5.5 tests=HTML_20_30,
     HTML_IMAGE_ONLY_04, HTML_MESSAGE, MIME_BASE64_ILLEGAL, MIME_HTML_ONLY,
     NO_REAL_NAME
X-Spam-Level: ****

<html><body> Dear user of <b>Anysite.com</b>,<br> <br> Your e-mail
account will be disabled because of improper using in next
<br>three days, if you are still wishing to use it, please, resign
your <br>account information.<br><br>

For details see the attach.<br>
```

[2]The details of this email and the subsequent investigation are sanitized to mask the IP addresses, domain names, and other information that may identify the involved parties.

```
<BR>Password - <img src="cid:mbqeovmear.bmp"><BR> <br>Best
wishes,<br>     The Anysite.com team    
           
<ahref="http://www.anysite.com">http://www.Anysite.com</a></body></html>
```

The email also contains two attachments. The first attachment is a 2KB bitmap image, named "mbqeovmear.bmp," that is used to display the number 86580 beside the password field in the email body. The second attachment, named "TextDocument.rar," is a 21KB RAR-compressed file.

The victim of this worm is a user running a fully-patched Windows XP computer. The suspicious email was noticed by the user when he was using his email client, Microsoft Outlook. Aware that Outlook unsafely opens emails, the user viewed the email in a web-mail session. The user did not open the attachments, nor did he view the email with the embedded HTML being interpreted.

The goals of this investigation are to understand the source of the email, its effects, and what possible effects it could have in the future. The goal is to complete a thorough investigation so that the victim can inform others about the email and how to handle it. In addition, the investigator must be certain that the email did not cause any harm. The investigator is allowed to spend about four hours investigating this email, since the effects of the email will not get worse in that amount of time.

4.2.1 Initial Analysis

This investigation did not require any data to be collected, aside from the email and its attachments. The investigator first made an initial survey of the email to determine whether it was a typical worm or a malicious attack directed at the email recipient. As any time-conscious investigator would do, the investigator first scanned the email using a leading commercial anti-virus program with virus definitions which were updated prior to the scan. The email, however, was not flagged by the anti-virus program, meaning that the email either does not contains a worm or contains a worm that is too new to be detected. The investigator attempted to open the password-protected RAR file using the supplied password, but the RAR file was irreparably damaged and could not be opened. The investigator then performed a series of searches using Google for any information on the attachments. Neither of the attachment searches produced any matches that would

signify that they were parts of a worm.

Since the initial analyses did not yield immediate results, the investigator then had to posit a hypothesis about the email. Several features of the email stand out:

- The return path address does not match the sender's address.
- The email provided the password to the password-protected RAR file.
- The email was delivered directly to a major ISP, i.e. no prior hops were made before it was sent to the user's email server.
- The grammar used in the email body was poor.

The initial macrocosm hypothesis should incorporate and connect all of these clues. Based on the user's history of previously not having received any worms, and that the investigation is not urgent, the investigator arrives at the hypothesis that the email is a directed attack on the user by an unsophisticated attacker. The altered hypothesis is stated as, "an unsophisticated user mass-mailed a threatening email directly from his ISP in order to harvest user accounts." This hypothesis explains why the emailed contained a seemingly valid return-path email address and was mailed from a major ISP, since no sophisticated user would allow such incriminating evidence to be disclosed. The hypothesis also explains the fact that the email spoofed an administrator account and apparently important and confidential material about the account violations was password protected. Finally, the poor grammar denotes an unsophisticated attacker, since many such attacks are performed by young or uneducated attackers.

The first premise for proving this attack is to trace the source of the email and try to verify its source, which, represented logically, is high-level pattern matching using a series of low-level pattern matches. The email originated from webhost2.net, so the investigator first performs a reverse-DNS lookup on that site. The results show that the site's IP address is 10.50.2.5. This IP address is then checked via a whois lookup using the arin.net website. The result of the whois lookup shows that the IP address belongs to a U.S.-based ISP, Big ISP. The investigator now knows that the email did originate from the U.S. More information about webhost2.net is gathered by visiting that site, which redirects the investigator to another domain name, smalltechpeople.com. This site belongs to a web

hosting, web design, and computer repair company located in the Midwest. Subsequent whois and reverse DNS lookups on smalltechpeople.com show that both of the domains are owned by the same company and have the same IP addresses. This information is helpful for understanding the origins of the email, but it does not provide any conclusive evidence.

The first premise is pursued further by investigating the email's return-path address, which appears to be a personal address. This is the use of a microcosm hypothesis, where the first premise of the microcosm hypothesis is the aforementioned high-level pattern. The domain of the address, doe-smith.com, is searched for using Google. Two results were found, both pointing to www.doe-smith.com. However, the web site is down, so neither link works. Using Google's cached link, the investigator is able to load the pages, and these pages are part of a married couple's homepage. The pages provide several useful pieces of evidence. Both of their full names are listed, as well as their location in the Midwest. Searches for Mrs. Doe-Smith's name produce no results, but searches for Mr. Doe-Smith produce a slew of information. The two salient pieces of evidence are links to his four web hosting companies as well as information about his computer security skills and certifications.

The first hypothesis has been weakened due to the possibility that the email originated from a security expert who owns numerous web hosting companies. There are several problems with this. First, no security expert would implicate himself in such a sophomoric fashion; a skilled attacker would know that providing information that is directly traceable back to oneself is unthinkable. Second, this man owns and operates many web hosting companies, so the likelihood of his email address being used by an attacker is high. Furthermore, there is a possibility that Mr. Doe-Smith is being framed by a web hosting competitor who wants to damage this man's reputation.

Even though the hypothesis has been weakened, the investigation continues investigating Mr. Doe-Smith to verify that he is not the culprit. The investigator visits his web hosting sites for further information. Of Mr. Doe-Smith's four companies, only one - the website for the company located in the Southwest U.S. - has been updated in the past two years. The other companies appear to be operational, but that assumption is simply based on the websites being accessible. Investigations into these companies show that no direct contact information - phone numbers, addresses, support personnel names - are listed on the site; only a web-based form is available for contacting the company. Further Google searches are performed using the companies' names as the search criteria.

Over fifty complaints about the companies' harboring of spammers were found, and two of these complaints discussed how Mr. Doe-Smith has been known to move to a different state every few months to avoid being contacted about his questionable business practices. Even more interesting than that discovery is the fact that Mr. Doe-Smith is an active member of the same 300-member organization to which the victim of the email attack belongs. This small, esoteric professional organization is comprised of people from around the world, but the victim has never communicated with or heard of Mr. Doe-Smith.

At this point, the claim that this email was a directed attack gains credibility. Mr. Doe-Smith has shown himself to be capable of at least supporting email harassment, if not committing it himself. Furthermore, there is now a link between the victim and Mr. Doe-Smith. However, this does not completely implicate Mr. Doe-Smith. He has too much security expertise to perform such an unsophisticated attack. In addition, he owns and operates numerous web hosting companies that provide email, so it is still possible that one of his customers used their account to perform the attack, or someone wished to frame Mr. Doe-Smith.

The investigation of the email header has now stalled, so the investigator now turns to the RAR file attachment, which would consist of trying to repair the file so that the password could be entered or a password cracker could be run on the file. Most file repair software, however, are not capable of completely correcting damaged files, and since the checksum of the file must be correct, any attempt to repair the file would most likely fail. The second option is to contact Mr. Doe-Smith directly to see if he could offer any clues about the incident. The investigator does not wish to pursue legal action, so this option would not jeopardize the investigation in that respect. Nonetheless, if the email was a directed attack against the victim, Mr. Doe-Smith may respond to the questions by attacking the victim and his network more aggressively.

A third option is to discuss the case with other security personnel. This is the only option that the investigator considered viable. An investigator, no matter how skilled, cannot always solve a case independently. The field of security is so vast and rapidly changing that no person can know every type of attack, system nuance, and network protocol. In this case, another person informed the investigator that the attack was in fact a worm that has several variants, known as the W32.Beagle.M@mm worm. The investigator then had to completely alter his hypothesis.

4.2.2 Reformulating the Macrocosm Hypothesis

With the news of the email being a new worm, the hypothesis must be altered to finish the investigation. This investigation's goals was to rigorously investigate the causes and effects of the email so that any such future emails can be prevented and the effects of this email are fully mitigated, so the investigation is not complete until these goals have been met. The investigator now knows that the email is most likely the "M@mm" variant of the W32/Beagle worm. In addition, the investigator knows that the email may have been sent from Mr. Doe-Smith's email account unknowingly because of the worm, and Mr. Doe-Smith may have had the victim's email address because of their mutual membership to the aforementioned organization. The new hypothesis is stated as, "Mr. Doe-Smith's computer was infected by the W32/Beagle.M@mm worm, which sent out the email to the victim and caused no damage since the worm caused no code to be executed."

The first premise in this hypothesis is that the worm originated from Mr. Doe-Smith's computer. The proof of this claim is the original analysis that showed that the worm originate from Mr. Doe-Smith's network, and the fact that a strong correlation between Mr. Doe-Smith and the victim exists. This analysis is not completely verified, since the investigator would need to contact the administrators of the email servers and contact Mr. Doe-Smith to confirm that the analysis is correct. there is a high probability, however, that this is the case, and the source of the worm is not critical in the case where only an isolated user is sent the worm. If the threat of the worm propagating a network existed, then the source of the worm would need to be discovered.

The second premise is that the email is in fact the W32.Beagle.M@mm worm. Security and anti-virus vendors began posting information on this variant on March 13, 2004. Symantec published a thorough analysis of this variant that provides the low-level details of the worm, which enabled the investigator to match the information about the worm with the email's contents [41]. This premise is supported by performing high-level pattern matching on the worm's signature. Symantec's write-up said that the worm has the following signature:

- The email is spoofed to appear to be from the server's administrator, and this name may be one of several names, including "support."

- The email header discusses the user's account, and one of those headers is "Notify about using the e-mail account."
- The body of the email is divided into five sections, each of which contains between one to ten different possible messages. The worm randomly selects one to use for each. Most of the sentences use poor grammar.
- The email attachment contains either a ZIP or a RAR file that matches a series of file names, one of which is TextDocument.
- The attached file is password-protected, and that password is provided in the form of a bitmap image file.

The signature provided by the write-up is compared to the email, and every element of the signature matches that of the email. At the time of the investigation, Symantec had not yet released the virus update, so the investigator had to manually scan the email. Since the signature matched exactly, the investigator concluded that the premise is correct: the email is the W32.Beagle.M@mm worm.

The final aspect of the hypothesis that needed to be proven is that the worm did not infect the system and cause any damage. Once again, the investigator checked the victim's system for the worm's signature. Symantec's analysis of the worm claims that the worm allows for remote access into compromised systems via TCP port 2556, and the executable code is called "winupd.exe" and is located in the "%System%" directory. If either of these conditions exists, then the system has been infected. The investigator looked at the victim's system's open ports by executing "netstat -a," and he finds that that port is not open. He then checks the victim's %System% directory by executing `"dir \windows\system32\*.exe"` and finds none of the files. Based on these findings, he concludes that the worm has not infected the victim's system.

The hypothesis has been proven because the investigator showed that the email did originate either from Mr. Doe-Smith's computer or his network, the email is the W32.Beagle.M@mm worm, and the worm did not infect the victim's system. The investigator has met the investigation requirements and should not continue to investigate other evidence, e.g. performing timeline analysis and analyzing other computers on the victim's network. Post-analysis actions can now be taken. These may include informing other users about this worm and updating virus definitions when anti-virus vendors release them.

4.2.3 Observations

This analysis is relatively simple and compact. Once the investigator realized that the email was a worm, the analysis became mostly a matter of confirming that the worm was the W32.Beagle.M@mm variant and that the system was not infected. The amount of data that was analyzed was significantly smaller than a full-system analysis. The bulk of the data was from tracing the source of the email using Google, and the significant findings were taken directly from Symantec's analysis. These facts imply that an *ad hoc* approach would have sufficed in this instance.

Despite the simplicity of the analysis, the process of explicitly listing a hypothesis and directing the analysis towards the proof of that hypothesis provided the investigator with a more thorough analysis of the possible cause. If the analysis did not initially generalize the cause of the incident, the analysis would still have begun in the same manner, whereby the investigator would have had to scan the email with anti-virus software and then begin investigating the email header. The email header analysis would follow a similar path to the analysis in this section; the main difference would be that the *ad hoc* analysis would be directed at solving the low-level matters of where the email came from rather than being directed at the high-level matter of why the email was sent to the victim. This difference is useful for the investigator because he can direct his analysis towards a high-level goal instead of allowing each clue to dictate what is analyzed next. The remainder of the analysis would be the same, since the worm analysis provides the investigator with all of the information he needs to finish the analysis.

The second advantage of this thesis' methodology is that the conclusions are already summarized in high-level forms both during and after the analysis phase. This inherent summarization feature allows the investigator to know what has been proven and to easily describe the state of the analysis to another investigator or interested party. Thus, if two investigators were working on the case together, they could easily share their ideas, analysis, and hypotheses with one another without having to delve too deeply into unnecessary detail.

The third advantage of this methodology is its effectiveness when data needs to be recollected during the analysis phase. In this investigation, the investigator had to reformulate his analysis hypothesis and collect new system data, i.e. information about open ports and files on the system. This data was suggested by the hypothesis, which

narrowed the type and amount of data to that which were specifically required, without extraneous information. This is an extremely small amount of data to collect, so even the use of an *ad hoc* approach would result in just these data being collected. However, in the case of a more complex analysis, the specific hypothesis would limit that data to precisely the data needed.

4.3 File System Analysis

The Honeynet Project, a group of security experts who research honeypots, holds a series of forensics challenges each year. These Scan of the Month Challenges (SotM) pit members of the security community against one another by having them analyze captured data and report their findings. These challenges range from the analysis of network packet captures to worm analysis to file system analysis. Each of the submissions is then reviewed and ranked according to completeness of the analysis, clarity of the report, and sophistication of the analysis techniques.

In October of 2002, the SotM was to analyze a floppy disk image for evidence of illegal activity [39]. The image was FAT12, an antiquated version of Microsoft's FAT file system that was originally used for the DOS operating system. FAT12 is now the file system used for all Microsoft-compliant floppies, so any floppy that is used by Windows is FAT12. The floppy used in the challenge was seized during the raid of a drug dealer's house. The investigation should turn up as much incriminating evidence as possible.[3]

In addition to reviewing and ranking the submissions, two leading forensics researchers published their own analysis reports. Their reports were supplied after the results of the competition were posted in an effort to provide less experienced forensics practitioners with models of proper analysis. This section explores the analysis techniques and methodologies from both of their reports. Each researcher's analysis is then juxtaposed against this thesis' methodology to illustrate the differences. From this examination, the strengths and weaknesses of this thesis' methodology for a low-level file system investigation are demonstrated.

[3]This investigation involves the capture of incriminating evidence, but the analysis presented in this section pertains only to the technical matter of uncovering and analyzing the evidence, not the proper techniques for evidence handling.

4.3.1 First Investigator's Analysis

Brian Carrier is a leading forensics researcher in the area of file system forensics. He is the developer of the TASK toolkit, which is a suite of forensics tools that runs under a number of Unix-based operating systems, including Linux, Solaris, FreeBSD, and OS X [8]. Carrier also published several journal papers and whitepapers while working at @stake, a leading security consulting company [43]. His file system expertise makes his analysis of the competition investigation noteworthy.

Carrier conducted all of analysis using his TASK toolkit and the Autopsy forensics viewer, which is an HTML front-end for TASK. He began his analysis by examining the file allocation table (FAT) for evidence of files on the system. The tool showed that the image had three FAT entries, with the corresponding filenames of "cover_page.jpgc," "Jimmy Jungle.doc," and "Scheduled Visits.exe." The FAT entries supplied the file size, inode number, and MAC times of each. In addition, "Jimmy Jungle.doc" was marked as deleted.

Carrier examined each file, beginning with "cover_page.jpgc." This file appeared to be a JPEG image, despite the extra "c" at the end of the file extension, but the file did not open inside of a JPEG viewer. Upon further analysis, Carrier discovered that its inode number, 8, is only allocated one sector. Since one sector is only 512 bytes and the file is supposed to be 15585 bytes, the file or the FAT entry must have been altered. Carrier decided to analyze the other files before returning to "cover_page.jpgc" in hopes that he would have a better understanding of the problem after subsequent analysis.

He next examined "Jimmy Jungle.doc." This was the file that had been marked as deleted. Carrier recovered the starting sector from this file's inode. FAT retained the starting sector for deleted files, but the FAT sector chain was set to zero. Carrier began with the starting sector, number 33, and calculated the number of sectors that would be needed for the document's file size, which are 40, so sectors 33-72 should be recovered. He verified that this was correct by examining the FAT entries, which showed that files were located in sectors 73-103 and 104-108, and since the file system began storing data at sector 33, the theory that the file was located in sectors 33-72 was correct. He then recovered sectors 33-72. After doing so, he opened the file using a Microsoft Word viewer, and this file was discovered to contain incriminating evidence.

Carrier next examined the third file, "Scheduled Visits.exe." He extracted the file's strings by running the UNIX program *strings*. This showed that the program was a zipped

file because of the message, "Zip archive data, at least v2.0 to extract," located inside of the file. After trying to unzip the file, Carrier received an error that the zip file was corrupt because use of the error, "the end was found." Carrier examined the FAT chain and discovered that sectors 104-108 formed a sector chain, but "Scheduled Visits.exe" was only listed as being 1000 bytes in its FAT entry. This meaned that the suspect had modified the FAT entry to only show the file to be 1000 bytes so that it could not be opened, and in addition to his attempt to conceal the file's contents by labeling the file with the ".exe" file extension. Carrier extracted sectors 104-108 and attempted to unzip the file. The file did not generate any errors, but it required a password for extraction. Carrier decided to return to "cover_page.jpgc" in hopes that the password would be located there.

The next step was to continue the analysis of "cover_page.jpgc." Carrier applied the knowledge of the sector chains to try to recover the file. This file's FAT entry points to sector 451, which was not allocated. Knowing that sectors 73-103 had been allocated and that the file size constitutes 31 sectors, Carrier extracted those sectors. As he did before, Carrier ran *strings* on those sectors to find embedded information. One such piece of information was "pw=goodtimes." He then opened the image file, which was more incriminating evidence.

Using the password "goodtimes" that was found inside a sector of "cover_page.jpgc," the zipped file was extracted. The file inside the archive was a spreadsheet that provided additional incriminating evidence. At this point, Carrier scanned the remaining unallocated sector for data, but found none.

4.3.2 Second Investigator's Analysis

Dan Kalil is also well known in the forensics community. He is an active member of the Digital Forensics Research Workshop, an organization that holds an annual forensics research conference and promotes research in computer forensics. Kalil has held positions in both forensics research and law enforcement.

Kalil used two commercial tools for his analysis, EnCase and Norton Disk Edit [27]. He began by noting the disk properties, e.g. the number of allocated bytes and number of bytes per sector. He then ran Disk Edit on the image, which reported several errors, and viewed the root directory information, which contained the information in the FAT. Noting the errors, Kalil analyzed the image with EnCase, a highly sophisticated forensics analysis tool used by the FBI and top forensics consultants. With the aid of EnCase, Kalil

was able to fix all of the errors noted by Norton. This allowed him to recover all three files. He was unable to open "Scheduled Visits.exe," but he knew that it was a zip file requiring a password. To uncover this password, he used EnCase's string search function to uncover all strings within the sectors, and he quickly found the password - which allowed him to unzip the file.

4.3.3 Observations

This investigation in the competition was highly technical and did not contain much complexity in terms of amounts of data to be analyzed. Most investigations require far more than a single floppy's worth of data to be analyzed. This investigation only involved about 50KB of data. The challenge of the investigation was for an investigator to find all of the evidence and be able to recover everything. The small amount of data coupled with the suspect's attempts to hide data posed either a test of an investigator's knowledge, as in the case of Carrier, or a test of his tools, as in the case of Kalil

This thesis' hypothesis is clearly not suitable for simplistic and automatable investigations such as this example. As Kalil's analysis demonstrated, forensics tools are able to recover forensic evidence quickly and effectively. Since Kalil knew which files existed, he could use EnCase to uncover the files. Carrier was able to perform his analysis by arbitrarily choosing which file should be analyzed first. He did not need to form a hypothesis about what had occurred, because the amount of data that was presented was minimal and could be analyzed completely. This thesis' methodology should not be applied in cases such as this, where the amount of data is small or the investigator needs to analyze all of the evidence. The methodology is only useful in cases where the investigator needs to minimize the amount of time spent analyzing and/or minimize the amount of data to be analyzed.

If this thesis' methodology were used for such an investigation, it would not provide any additional information or convenience, and it most likely would prolong the analysis unnecessarily. In this case, the initial hypothesis was that the disk contained incriminating evidence that was intentionally hidden by the suspect. The analysis would begin by proving the intentionally hidden aspect using a microcosm hypothesis that analyzed each file in turn. Therefore, the microcosm hypothesis would be divided into three smaller microcosm hypotheses, each being proven by high-level pattern matching. The macrocosm's next premise was that the disk contained incriminating evidence; this premise is proven by

using high-level pattern matching that uncovers the evidence contained within each file. Therefore, the use of such proofs does not yield any benefits to the investigator. The analysis would proceed naturally without the use of such proofs, as demonstrated by Carrier and Kalil's analyses.

4.4 Investigation from Forensics Literature

The investigation in this section is taken from Hacker's Challenge, a collection of twenty forensics investigations performed by upper-echelon security professionals [35]. The book is structured as a series of twenty challenges. The reader is given details about the discovery of the incident, i.e. the event(s) that alerted a system administrator or user that a security incident had occurred. The reader is also provided with the details of the data collection and initial analysis. Most of this information is provided in the form of log records. The reader is then presented with a series of questions that require him to correlate the data to form a complete analysis report about the causes and effects of the incident. The author, Bill Pennington, provides the answers with justifications at the end of the book.

This section presents the details of the analysis phase of an incident. Pennington provides a detailed account of the investigation, starting from the discovery of the incident to the final stages of analysis. This account yields insight into the progression of a sophisticated forensic investigation that is comparable to the methodology put forth in this thesis. The story, however, does not discuss the underlying motives and thought processes of the investigators. As such, the details of the investigators' thought processes are presented throughout the investigation as they pertain to this thesis' methodology. The final subsection summarizes the basic methodology used by the investigators and how this thesis' methodology compares.

4.4.1 Background

On a Monday morning, Kris, a system administrator for a medium-sized company that runs Windows 2000 and a Microsoft Exchange email server, began receiving complaints from the company's employees that they were missing some or all of their emails. Kris

checked the email server for problems, found nothing wrong, and rebooted the server. By the end of the day, Kris had received complaints about the problem from fifty users. Kris checked that the Exchange server was fully patched. It was, so the incident was not the result of a well-known security vulnerability. It was then that Kris realized that the fifty users who had experienced email problems were some of the most important company employees.

The next morning, Kris received word from another I.T. staff member who said that employees were completely unable to connect to the email server. When Kris examined the server, he discovered that the entire Exchange database had been deleted, which meant that every user's email had been deleted. Kris restarted the server and restored the email from backup storage. At this point, Kris realized that the incidents were related and most likely the work of an insider, so he contacted forensic investigators to handle the incident. Upon contacting the investigators, an email was sent out to every employee, describing how the attacker had compromised the company's web server two weeks prior and had decided to send a stronger message by compromising the email server as well. The attacker did not attempt to extort the company; he only said that the company should have a more competent I.T. staff. To substantiate his claims, he sent out the passwords to several of the high-ranking employees' accounts and also the administrator password. The IP address of the email matched the IP address of the company's first-hop gateway, which meant that the email was sent from inside the company's network.

Pennington noted that the company's outward-facing website had been the victim of defacement two weeks prior. The defacement was quickly fixed. However, the company was embarrassed by the incident. A small hacker group from France had taken credit for the defacement on attrition.com, a website that links to website defacements and hosts a message board for hackers to claim the defacements. The French hacker group was not well known and had a history of only performing trivial, ready-made, script-based attacks.

The investigators began their initial analysis by forming a timeline of events. They interviewed Kris about the times and natures of the events. While interviewing Kris, an employee reported that all of his email was again missing; more employees reported this problem throughout the rest of the day. The investigators agreed that the attack appeared to be a company insider, so they formed a list of suspects and began collecting logs. The logs they were most concerned with were the physical security logs, firewall logs, Windows 2000 event logs, and virtual private network (VPN) logs.

4.4.2 Incident Analysis

The investigators had a set of significant clues to consider. The email incidents affected only the most important company employees; non-executive and non-managerial employees were not affected. The email that was sent to the company originated from inside the company's network. So the attacker had considerable knowledge of the company structure and access to the network. The web site defacement seems to imply that the same group carried out the email server attack, but based on the French hacker group's skill, that is unlikely. It was possible that the company's security vulnerabilities were exposed by the defacement. Another important piece of evidence was that the Exchange server had already been fully patched, so a trivial attack could not have granted a hacker access to it.

The investigators believed the attack originated from someone internal to the company and not an external hacker, such as the hacker group that had defaced the web site. This was because the defacement was traced backed to the French hacker group one week after the incident; the group did not demonstrate that they had the skills to perform the Exchange sever attack, and they did not fully penetrate the company's network when they defaced the website. By investigating in this manner, the investigators tacitly formed a hypothesis, and they directed their investigation towards proving that the attacker was internal to the company by determining who had access to the network at particular times. The timeline foci were the times around which the Exchange server behaved anomalously.

The questions that are posed to the reader, i.e. the questions that the investigations had to answer, involve the users that were logged in during the times when email was deleted, which users were connected via VPN, what IP addresses belonged to the logged-in users, and whether any unusual behavior observed during those times. While these questions are tightly coupled with timeline analysis, they do demonstrate that the investigators were approaching the case in a formal manner. They systematically wished to pinpoint the internal user that performed the attack. In addition, they also wanted to find anomalous behavior around the incident times to substantiate their hypothesis or refute it. That is, they actively searched for a counterexample.

The investigators discovered that Chris Miller, a marketing employee, logged in under his user account "cmiller" via VPN just before the incident and logged out immediately after the incident concluded. Chris was a marketing employee, so the investigators analyzed further, because the investigators doubted that a marketing employee could per-

form such an attack. They discovered from the physical security records that Chris entered the building at the same time that he was logged into the network via VPN, and they were informed that Chris did not own a home computer. They thus suspected that Chris's account was compromised by the attacker. They refined their hypothesis to hone in on activity related to Chris's account breach. They scanned logs for the IP address that was used when Chris's account was used for the attack. The scan uncovered that the IP address was used in prior attempts to log into other users' accounts, those user accounts which were some of the company's top employees. This focused attack further substantiated the hypothesis that the attack was an inside job.

The investigators were able to find the employee whose IP address matched that of the attacker's by scanning the VPN logs for matching IP addresses that were not hacking attempts on other users' accounts. The user turned out to be Scott, a seemingly trustworthy I.T. staff member who had little apparent reason to perform the attack, but who did possess the technical aptitude for such an attack. While Scott appeared to be a benign employee, he matched the investigators' hypothesis of a company insider who possessed enough technical knowledge to perform the attack. The investigators questioned Scott, and the investigators were allowed to make images of his home computers.

The investigators analyzed the images of Scott's hard drives and volatile data with the goal of uncovering evidence that related to Chris Miller and other upper-level employees, the times of the incidents, and general hacking tools such as password crackers. They were unable to uncover any significant evidence by examining the file system for suspicious files. This suggests that Scott did not use these computers for the attacks, or he took steps to delete incriminating evidence. The investigators' hypothesis was strengthened, however, when they analyzed Scott's temporary files, virtual memory, and slack space and discovered the following:

- His web browser's temporary files revealed that he had visited several websites pertaining to hacking;
- Numerous company emails were found in the virtual memory regions, yet none of these emails were to or from Scott;
- The output from a cracking program containing over 400 usernames and passwords

of the company's employees was found in the slack space of the hard drive, which signified that he had deleted this information;

- Several hacking programs were also discovered in the hard drive's slack space.

The investigators concluded their investigation as this point due to the overwhelming evidence against Scott. Their hypothesis that an employee with technical proficiency had performed the attack was proven. Scott used his home computers to run a password cracker against the company's user accounts. He then connected to the company's VPN using the cracked employee accounts. From there, he used his insider knowledge to target upper-level employees' accounts. He downloaded others' email and then deleted specific employees' email. He then increased the severity of the attacks by deleting the entire Exchange database. When he received word that investigators were searching for the culprit, he naïvely attempted to destroy the incriminating evidence by deleting the files using the operating system's deletion mechanism without attempting to remove the remnants of those files residing in used space on the hard drive.

4.4.3 Observations

The investigators followed a strict, organized plan from the initial stages of their analysis. They made an assessment of the incident and the surrounding circumstances. Based on that assessment, the investigators formed an investigatory plan that they consistently followed. Every step they took was directed at substantiating that plan. This direction provided a clear investigation that did not involve extraneous analysis. In addition, the investigators ensured that the plan was correct by correlating every piece of analysis and checking for anomalous data that contradicted their analysis.

The underlying analysis methodology used by the investigators corresponds with this thesis' analysis methodology. The investigators' plan is used in the same manner as a macrocosm hypothesis. Initial analysis is examined to formulate a basic theory about the causes and effects of an incident. That theory is then refined as further analysis is performed. More strikingly, the investigators followed the plan in a premise-based manner. They did not perform the path-based analysis; instead, they stuck to a single premise and analyzed according to that. For example, they did not attempt to gather every piece of evidence pertaining to Chris Miller's account usage. They focused on finding the internal

user who attempted to attack the Exchange server, a process consisting of finding the IP address that was used by the attacker. In other words, the investigators did not stray from their initial premise by discovering all they could Chris Miller's account; they pursued the premise that the attacker was technically savvy and would not use his own account to perform the attacks. This directedness at every stage of their investigation represents the methodology that this thesis espouses. The only difference is that the investigators did not explicitly discuss their thought process or analysis methodology.

While the investigators demonstrated excellent analysis skills, the incident did not produce significant hypothesis modifications. The initial analysis led the investigators to the relatively obvious conclusion that the incident was the work of an insider. The question remains about how the investigators would have responded in the case of the attack being an outsider that was able to gain insider access. The methodology put forth in this thesis adequately handles such situations, but this incident does not demonstrate the effectiveness of the investigators' methodology.

4.5 Conclusions

This chapter demonstrates the workings of this thesis' methodology in an assortment of forensic investigations. Each investigation discussed had unique requirements, such as the degree of completeness or urgency of completion. Each investigation also varied in terms of the amount of evidence available and the number of possible causes and effects. These factors, when present in actual investigations, reveal the workings of this thesis' methodology. This section reexamines how the methodology fares in these case studies and accordingly generalizes about the methodology.

4.5.1 Advantages of the Formal Methodology

The formalized methodology is designed to mitigate the effect of analysis complexity on the investigator. The case studies in this section verify this. The first investigation, involving the hacked Windows XP system, was a fairly complex investigation. The hacker took careful steps to cover his tracks, thwarting simplistic forensic analysis. This thesis' methodology helped organize the ideas of the investigator so that fruitless or hasty analysis

was not performed, such as immediately switching from analyzing the occupied hard drive space to analyzing network logs without finishing the hard drive space issue. In the case of the email worm, the investigator was forced to reformulate his hypothesis, but because of the high-level structure of his analysis, he was able to do so while incorporating the prior analysis into the new hypothesis. Without the use of this methodology, the organization of the prior analysis would require that the reformulation take more effort on the part of the investigator. Finally, the case study of the Exchange server compromise demonstrated that structured analysis that roughly approximated this thesis' methodology provides an effective and efficient means for performing and concluding analysis. These cases are both excellent illustrations for the type of investigations where this methodology is best suited.

This methodology works best in cases where the complexity of the case is sufficiently high that an investigator may have difficulty in properly structuring his analysis. Each of the cases offered a multitude of analysis paths. With the methodology, however, these analysis paths more made clearer, and the analysis did not stray from the goals of the analysis. This methodology thus provides an investigator with a means for paring down the complexity of investigations into more manageable, abstract units, thus enabling the investigator to direct his analysis in a more logical manner.

4.5.2 Disadvantages

The striking example of this methodology's failing is the low-level analysis of the confiscated floppy disk. This example demonstrates that if the case is not sufficiently complex, the methodology not only does not provide additional usefulness, it prolongs the investigation. The analysis required was mostly low-level tasks that could be performed almost entirely by tools, as shown by Dan Kalil. Since investigations such as this do not require a hypothesis or other analysis path direction, the methodology provides no benefit. Moreover, the use of proofs requires the investigator to perform actions that are extraneous and time consuming.

A second disadvantage is that this methodology may not provide benefit even in a complex case. Consider the first case study, the hacked Windows XP system. Suppose the investigator just wanted to rid the machine of the backdoors and other planted files. If he were knowledgeable of Windows XP, he may be able to uncover said files without the use of a hypothesis. Since he does not need to report his findings to others and is not interested in

finding the exact cause of the incident, the use of proof-based analysis does not help him. The methodology is not useful in cases where an investigator does not need to report his findings or be complete. However, these factors reduce the complexity, which causes the investigation to be relegated to the category of a uncomplex investigation.

Chapter 5

Conclusion

Computer forensics is a relatively new field of study. While the body of literature for forensic science has been well developed, the same cannot be said of computer forensics. A survey of computer forensics literature shows that only a handful of books and journals are devoted to this field of study. Compare this with the other areas of computer security - such as intrusion detection and cryptography - which already have a strong foundation of literature. Obviously, the issues surrounding computer forensics have not been fully explored.

Forensics involves numerous phases, including data collection and incident recovery, but analysis is the backbone of all forensics work. No facet of computer forensics is as critical as analysis. Without proper analysis, the purpose of performing an investigation is lessened, if not altogether defeated. Every investigation should apply proper analysis techniques and methods, and these techniques and methods must coherently work together for the analysis to be effective and efficient.

The majority of current computer forensics research focuses on technical, jurisprudence, or administrative issues. The issues that the literature addresses pertain to conducting analysis for a specific type of case or performing a certain type of technical analysis. While all of these issues are important, they are directed at solving immediate, practical problems; that is, they do not address underlying theoretical issues. The theoretical issues may be addressed indirectly by the technical focus, as in the model of investigative phases. The main focus of the literature remaining the immediate solving of a specific, practical

problem.

Because of the dearth of rigorous academic computer forensics research, the techniques and methods used by practitioners tend to be applied in an *ad hoc* manner. Most computer forensics literature is produced by practitioners who devise techniques and methods to be used in specific situations. These techniques are rarely tied to an underlying methodology. An investigator thus uses a hodgepodge of these techniques, and he applies them without an underlying methodology. This *ad hoc* application of the techniques obstructs a truly standardized analysis framework from being employed.

Computer forensic investigations often involve critical situations where the analysis must be rigorous and standardized. Law enforcement investigations must be able to prove that their conclusions from a case are valid, and system administrators and emergency response investigators need to be able to quickly and accurately determine the cause and effects of an incident and correct them accordingly. Both of these investigation types require that the investigator be able to respond to every case in an appropriate manner. While the use of known computer forensic techniques may be suitable for solving cases from a technical standpoint, they do not provide the investigator with a standardized means for organizing the investigation. Since both of the cases require that an investigator be able to quickly and effectively respond to an incident, the investigation should be standardized to minimize logical oversights and provide coherently organized analysis.

5.1 Summary

This thesis presents a formal methodology for forensic analysis that serves as a framework for structuring the analysis phase of investigations and provides a natural means for performing other investigative phases. This methodology structures the analysis findings as a general proof, called the macrocosm proof. Each of the macrocosm proof's premises is itself a proof, albeit of a more technical nature. The proofs are structured such that the maximum amount of abstraction occurs, so that the high-level proofs are very conceptual, and the technical proofs are highly technical. This proof can thus be viewed as a tree, with each level of the tree corresponding to some level of abstractness, e.g. the leaf nodes are the most technical and the root is the macrocosm hypothesis.

The methodology makes use of four types of proofs that are founded on logical

techniques from related fields. The macrocosm hypothesis and microcosm hypothesis proofs contain a single hypothesis that includes the main causes and effects of an incident, and it is proven by a series of premises that are drawn from the conclusions of more technical proofs. These inductive proofs are similar to those of artificial intelligence, where an initial data set is analyzed, a hypothesis is formed from that analysis, and the hypothesis is then proven. The main similarity is that artificial intelligence uses techniques to cope with learn and adapt while proving the hypothesis; this occurs during computer forensic analysis, since unknown evidence is being examined. The high-level and low-level pattern matching proof techniques are derived from data mining and anomaly detection. Both of these fields search for a known pattern and abstract information from the findings to form a new finding.

The methodology offers numerous benefits to investigators, but those benefits only apply to certain types of investigations. Case studies demonstrate that the methodology is highly adaptable and formats the analysis from complex investigations into a more understandable and manageable form. This methodology is excellent for complex incident response cases and computer misuse. It is not, however, well suited for investigations that do not involve a high degree of complexity or a large set of data to be examined. The case involving the seized floppy disk of a suspected drug dealer confirms this drawback. Since simple cases do not require that a large case be tied together in a logical fashion, an investigator is not faced with the data reduction problem, whereby he must juggle analysis completeness with the mitigation of overwhelming complexity. Therefore, the methodology is disadvantageous because of the extra time and effort that must be spent structuring the investigation in such a manner.

5.1.1 Contributions of this Thesis

The main problem facing computer forensics research is not the difficulty of solving technical problems, such as quickly and effectively uncovering a trojanized system. Rather, the problem is that a dearth of foundational research has been conducted. The research has typically focused on solving immediate problems instead of long-term problems that require a robust, formal body of literature. For computer forensics to develop into a well-respected and effective field of study, the literature must establish what constitutes effective investigative practices. The foremost issue is that of analysis. Analysis cannot be measured simply by the conclusions from that analysis. There must be additional criteria, such as

the structure of the analysis and how well the analysis responded to unforeseen evidence.

This thesis provides several contributions to computer forensics literature. The immediate practical value of this work is the benefits of applying this methodology. These benefits are the logical and more coherent structure of the analysis, the adaptability of the analysis to unforeseen clues and changes to the analysis plan, and the explicit requirement that every step during analysis be directed at solving a single problem. In addition, this methodology restructures the four-phase model of investigations to allow for returning to previous phases from the analysis phase without the need to use an iterative version of the four-phase model. That is, an investigator may continue to use the effective four-phase model, but because the four-phase model is too rigid to handle returning to one phase without difficulty, this methodology improves upon that by structuring the analysis so that it more naturally may be restructured after performing additional steps in a previous phase.

The theoretical contributions of this thesis are both the standardization of analysis and the hopeful inciting effect of formalizing an aspect of computer forensics. The analysis methodology is logically sound. Based on ideas and techniques from related fields, it is shown to be effective in the case studies. As such, this methodology is an improvement over the existing *ad hoc* analysis approach. In addition, no such methodology is present in computer forensics literature, whether it is for analysis, data collection, or the entire investigation. Since no such methodology exists, this methodology should encourage researchers to continue formalizing and revising other areas of computer forensics. Researchers should not be satisfied with the "black artstatus of computer forensics that seems to prevent such formalizations.

5.1.2 Case Study Results

The case studies demonstrate the relative advantages and disadvantages of this methodology, but most importantly, they show that every step during analysis may be formalized and consistent with a general methodology. The incident response cases and email worm analyses are examples of the strengths of the methodology. Each of these cases possesses a high degree of complexity and/or uncertainty. In the case of the hacked Windows XP machine, the system was cleansed and trojanized to cause it to purport incorrect information. While the techniques employed to uncover the causes and effects are not novel, the organization of the information and directedness of each step produced

clear analysis. At every step, the investigation proceeded along a known path, rather than randomly exploring various paths until the case was solved.

The disadvantage of this methodology was demonstrated in the seized floppy disk investigation. The goal of the methodology is that analysis should be structured so that an investigation is better able to cope with complexity. The floppy disk case runs counter to this goal, since the investigator is not faced with high complexity. The resulting examination of the analysis proves that there are limitations to which what types of cases this methodology may be applied.

5.2 Future Research

The proof-based methodology offers a novel means for viewing computer forensic analysis. Best practices for computer forensics claims that every step taken during the investigation be documented. The documentation is a combination of logging every command entered on a system and pencil-and-paper documentation of the additional steps that were not logged on the system, such as the times at which an investigator spoke with system administrators and users about an incident. This methodology offers an additional level of documentation that provides high-level and technical details about the direction of the investigation, an investigator's thought process, and assumptions about data that does not require analysis. Because the proof-based methodology forces an investigator to explicitly document his thought process, this information provides additional clues and more thorough documentation. This novel view of the analysis process can be utilized to create several useful tools for improving the analysis process.

5.2.1 Automated Analysis

The first area of research to be explored further is perhaps the holy grail of computer forensic analysis: analysis automation. Completely automated analysis is perhaps not possible, since most computer forensic investigations involve human factors, e.g. a company's network was compromised one week after their system administrator was fired. Semi-automation, however, is possible. The analysis can be divided into logical elements based on the macrocosm hypothesis, and each of these units may be solved via tools that are

given a set of criteria for solving the premise and the set of data over which it performs the analysis. This type of semi-automation can proceed in a similar fashion to theorem provers or model checkers. The main difficulty for semi-automation is defining the criteria by which the premise is said to be proven. In computer forensics, data is often highly volatile and evidence may be intentionally hidden or destroyed in cases involving a malicious suspect, so the semi-automation must be given precise criteria about what constitutes a proof and how to handle contradictory evidence. The investigator can feed the semi-automation tool with the criteria based on his suspicions of the incident and whether the suspect may have attempted to deceive the investigator by hiding or destroying incriminating evidence. The main difference between the theorem prover and model checking paradigm is thus that the proof must correspond to a statistical value and not an unambiguous objective proof or refutation. A computer forensic analysis semi-automation tool will become increasingly valuable as the size of disk storage and the number of networked devices increases, since these factors will increase the amount of data and the difficulty of and time required for analysis.

5.2.2 Data Management Tool for Investigators

A more immediate tool and research topic stemming from the proof-based methodology is that of a data management tool that an investigator uses to track the progress of his analysis. The "pencil-and-paper" approach for documenting analysis is arcane and cumbersome. An investigator can certainly use spreadsheets, diagrams, and other reporting tools to document the analysis, but these formats still do not offer a clear, intuitive representation of the analysis without careful consideration. With the proof-based methodology, a documentation tool can present the analysis in a convenient form. For instance, the tool could display the analysis on varying levels using a hierarchical flowchart, e.g. a IDEF0 flowchart. The macrocosm hypothesis is both displayed at the top level, with each of its premises below it. The tool would allow for simple additions of new premises and hypothesis reformulations. It could also display previous analysis that is deemed irrelevant to the hypothesis. This tool would be extremely valuable for analysis reporting and visually understanding the analysis structure. Furthermore, it would further aid an investigator with documentation, since the documentation would be centrally located and no significant thought would be required of an investigator about structuring the documentation.

5.3 Concluding Remarks

Computer forensics, as a practice, is a field that demands precision. Every investigation is critical, so an investigator must be precise with every case. Since computer forensics is effective, the investigative techniques must be effective. If the techniques were ineffective, then the new techniques would be required, or computer forensics would be a complete failure. Computer forensics is thus not in danger of failing entirely. As long as an investigator is able to come to the proper conclusion, then computer forensics is succeeding.

The notion of succeeding at an investigation, however, is misunderstood in computer forensics. A computer forensics investigation is viewed as successful if the correct conclusion is found within the proper time frame. For example, in the case of the hacked Windows XP machine, the investigation is considered successful if the cause and effects of the incident are discovered. This definition of successful does not take into consideration factors such as the organization of the conclusions and the corresponding evidence, order of evidence analysis, and the thought process behind the analysis. This last factor is especially important when an investigation is handled by a team of investigators or in situations where an investigation must testify in court about his analysis. A detailed log of every step taken during analysis and the thought process of each may be useful, but without any structure to the order of the steps, the thought process may appear convoluted and perhaps questionable. One must always remember that not every computer forensic investigation involves straight-forward, logical analysis. Investigators must be able to make logical leaps, whereby the justification for such a leap is not entirely logical.

A proof-based methodology may seem extraneous, but it offers several benefits that make it immediately valuable and potentially more valuable as computer forensics develops. The argument may be made that all skilled investigators tacitly form proofs when they thoroughly analyze and either organize the analysis immediately or organize the results of their analysis when they report the findings. This argument may be correct in that skilled investigators logically perform analysis. The argument fails, however, to show why all investigators cannot make use of the methodology. This methodology could be applied by any investigator, skilled or not. Even skilled investigators may not always tacitly perform their analysis in a structured, proof-based manner. This methodology makes the logical component explicit, thereby making an investigator mindful of the logic of the analysis. Moreover, even the most skilled investigators may not be able to properly communicate

their findings to other investigators. This methodology facilitates clearer communication by organizing the findings into high-level structures that are easily understood, and from those high-level structures, the technical details can be understood by examining the technical details of each.

This methodology offers many benefits that are of immediate benefit to investigators, but the true value is it serves as a basis for further research into the understanding and formalizing of computer forensics as an academic field of research. This thesis discusses the many theoretical gaps of computer forensics, the benefits of this methodology, and future applications of proof-based analysis. With the burgeoning of computer forensics in academia, the proper foundations must be established upon which future discussions make take place. This thesis provides the beginnings of such foundations by highlighting the procedures and inherent difficulties in computer forensics. The four-phase model of an investigation is too rigid for proper investigations; even an iterative version of that model is inappropriate. This methodology is a more natural method for performing analysis, since it easily allows an investigator to go back to previous phases while mitigating the cumbersome task of plan reformulation. Moreover, the methodology provides a definite structure to all investigations - one that is based on logic. The proof-based methodology augments computer forensics literature with a formal methodology that is similar to those found in well-developed fields, such as software engineering and artificial intelligence.

Bibliography

[1]

[2] 4Law. FBI: Moussaoui e-mail not recoverable 1/1/03, January 2003. Available from URL ttp://www.4law.co.il/L101.tml.

[3] Evert Beth. *The Foundations of Mathematics*, page 741. Harper and Row, NY, 1966.

[4] Matt Bishop. *Computer Security: Art and Science*, pages 727 – 738. Addison-Wesley Professional, Boston, MA, 2002.

[5] M. Bongard. *Pattern Recognition*, pages 6 – 10. Spartan Books, NY, 1970.

[6] Brian Carrier. Defining digital forensic examination and analysis tools. *In Digital Research Workshop II*, 2002.

[7] Brian Carrier. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of Digital Evidence*, 1(4), Winter 2003.

[8] Brian Carrier. The sleuth kit & autopsy forensic browser, 2004. Available from URL http://www.sleuthkit.org.

[9] Eoghan Casey. Error, uncertainty and loss in digital evidence. *International Journal of Digital Evidence*, 1(2), Summer 2002.

[10] Microsoft Corporation. Fat: General overview of on-disk format. 1.03, December 2002.

[11] DMZS. F.i.r.e. forensic and incident response environment bootable cd, 2004. Available from URL http://fire.dmzs.com.

[12] Mark Reith et al. An examination of digital forensic models. *International Journal of Digital Evidence*, 1(1), Fall 2002.

[13] Dan Farmer and Wietse Venema. Computer forensics analysis class handout, 1999. Available from URL `http://www.fish.com/forensics/class.html`.

[14] Dan Farmer and Wietse Venema. TCT, 1999. Available from URL `http://www.fish.com/tct/`.

[15] Technical Working Group for Electronic Crime Scene Investigation. Electronic crime scene investigation: A guide for first responders. July 2001. Available from URL `http://www.ncjrs.org/pdffiles1/nij/187736.pdf`.

[16] Richard C. Froede, editor. *Handbook of Forensic Pathology*, page 20. College of American Pathologists, Northfield, IL, 1990.

[17] S. Garfinkel. Network forensics: Tapping the internet, April 2002. Available from URL `http://www.oreillynet.com/pub/a/network/2002/04/26/netapp.html`.

[18] Christoffer Gefwert. *Wittgenstein on Mathematics, Minds and Mental Machines*, page 105. Ashgate Publishing, Brookfield, VT, 1998.

[19] Susan Haack. *Philosophy of Logics*, pages 120 – 122. Cambridge University Press, NY, 1978.

[20] Aaron Hackworth. Dcomexpl unixwin32: Windows rpc dcom buffer overflow exploit, 2003. Available from URL `http://www.giac.org/practical/GCIH/Aaron_Hackworth_GCIH.pdf`.

[21] M. Hanna. Farewell to waterfalls. *Software Magazine*, pages 38 – 46, May 199.

[22] John Horgan. The death of proof. *Scientific American*, pages 91–103, October 1993.

[23] Feng-Hsiung Hsu. *Behind Deep Blue : Building the Computer that Defeated the World Chess Champion*, page 3. Princeton Publishing, Princeton, NJ, 2002.

[24] Warren G. Kruse II and Jay G. Heiser. *Computer Forensics: Incident Response Essentials*, pages 3 – 18. Addison-Wesley, Boston, MA, 2002.

[25] Edward Iwata. Enron case could be largest corporate investigation, February 2002. Available from URL http://www.usatoday.com/tech/news/2002/02/19/detectives.htm.

[26] Albert J. Marcella Jr. and Robert S. Greenfield. *Cyber Forensics: A Field Guide for Collecting, Examining, and Preserving Evidence of Computer Crimes*, page 14. Auerbach, Boca Raton, FL, 2002.

[27] Dan Kalil. Honeynet project scan of the month 24, October 2002. Available from URL http://www.honeynet.org/scans/scan24/sol/kalil/.

[28] Ron Kohavi. A study of cross-validation and bootstrap accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 14:1137–1143, 1995.

[29] E. Kounalis and P. Marquis. Fundamental methods for horn logic and artificial intelligence applications. *Proceedings of the 31st Annual ACM SIGUCCS Conference on User Services*, September 1992.

[30] Kevin Mandia and Chris Prosise. *Incident Response: Investigating Computer Crime*, pages 16–21. McGraw-Hill, Berkeley, CA, 2001.

[31] Eric A. Meyer. An interview with mike davidson of espn: Part 1, March 2003. Available from URL http://devedge.netscape.com/viewsource/2003/espn-interview/01/.

[32] Klayton Monroe and Dave Bailey. System baselining: A forensic perspective. 2003. Available from URL http://ftimes.sourceforge.net/Files/Papers/baselining.pdf.

[33] Gary L. Palmer. Forensic analysis in the digital world. *International Journal of Digital Evidence*, 1(1), Spring 2002.

[34] Cyrus Peikari and Anton Chuvakin. *Security Warrior*, pages 409 – 422. O'Reilly, Sebastopol, CA, 2004.

[35] Bill Pennington. The insider. *Hacker's Challenge*, pages 9–34 and 203–208, 2001.

[36] Brian K. Porter. Rpc vulnerability & exploit, 2003. Available from URL http://www.giac.org/practical/GCIH/Brian_Porter_GCIH.pdf.

[37] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*, pages 29–30. McGraw-Hill, NY, 5th edition, 2001.

[38] Paul E. Proctor. *The Practical Intrusion Detection Handbook*, pages 15 – 16. Prentice-Hall, Upper Saddle River, NJ, 2001.

[39] The Honeynet Project. Scan of the month 24, October 2002. Available from URL `http://www.honeynet.org/scans/scan24/`.

[40] Richard L. Rollason-Reese. Incident handling: An orderly response to unexpected events. *Proceedings of the 31st Annual ACM SIGUCCS Conference on User Services*, September 2003.

[41] Heather Shannon. Symantec security response: W32.beagle.mmm, March 2004. Available from URL `http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.m@mm.html`.

[42] Joe Sremack. Cross-validation of file system layers for computer forensics. *Digital Forensics Research Workshop*, August 2003. Available from URL `http://www4.ncsu.edu/~jcsremac/research/sremack_dfrws2003.pdf`.

[43] @stake. @stake research reports, 2004. Available from URL `http://www.atstake.com/research/reports/`.

[44] Tye Stallard and Karl Levitt. Automated analysis for digital forensic science: Semantic integrity checking. *19th Annual Computer Security Applications Conference*, 2003.

[45] Peter Stephenson. Modeling of post-incident root cause analysis. *International Journal of Digital Evidence*, 2(2), Fall 2003.

[46] John Tan. Forensic readiness, July 2001. Available from URL `http://www.atstake.com/research/reports/acrobat/\atstake_forensic_readiness.pdf`.

[47] the grugq. Defeating forensic analysis on unix. *Phrack*, 59, July 2002. Available from URL `http://www.phrack.org/phrack/59/p59-0x06.txt`.

[48] Tripwire. Tripwire homepage, 2004. Available from URL `http://www.tripwire.org`.

[49] CERT/CC. "CERT/CC Statistics 1988-2003", 2003. Available from URL `http://www.cert.org/stats/cert_stats.html`.

[50] John R. Vacca. *Computer Forensics: Computer Crime Scene Investigation*, page 4. Charles River Media, Hingham, MA, 2002.

[51] De Velopment. Summary: Security issues, purchasing a new, pre-loaded, windows xp computer. *SecurityFocus HOME Mailing List: FOCUS-MS*, October 2002. Available from URL `http://www.securityfocus.com/archive/88/294260`.

[52] Christopher Westphal and Teresa Blaxton. *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, pages 5–6. Wiley Computer, NY, 5th edition, 1998.

[53] xshadow. Vanquish rootkit, 2003. Available from URL `http://www.rootkit.com/newsread.php?newsid=35`.

[54] Alec Yasinsac and Yanet Manzano. Policies to enhance computer and network forensics. *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, June 2001.

Appendix A

"Cross-Validation of File System Layers for Computer Forensics"

A.1 Abstract

This paper[1] discusses the problem of quickly and accurately identifying a discrepancy between file system layers using semi-automated tools that provide useful clues via cross-validation of multiple layers. Using an illustrative example of a FAT12 file system, a basic cross-validation method for forensic investigation is discussed. By doing so, several inherent problems to cross-validation are shown, particularly how one abstract layer's representation does not neatly convert to another layer simply by using the input/output relationship. Nonetheless, the idea of cross-validation is ultimately shown to be promising in its ability to provide validated evidence.

A.2 Introduction

One of the major difficulties facing computer forensic science is the audit reduction problem, in which the sheer amount of information can render an investigator's task nearly

[1]The author presented this paper at the Digital Forensics Research Workshop in 2003.

impossible [17]. Such complexity can be drastically reduced by focusing on information at a higher, more abstract level. Abstraction allows an investigator to more quickly understand what has happened to the system, so he can present an account of the incident to a company supervisor or a jury. However, the benefits of using different levels of abstraction come at a price that a forensic investigator cannot ignore. The more abstract the data is, the more likely information will be hidden or lost [9]. Because every layer of abstraction uses its own form of meta-data, information will inevitably be lost if every layer is not examined [7].

The method proposed in this paper for producing validated evidence in a practical manner is the cross-validation of file system layers. Cross-validation requires that high-level pieces of evidence be compared to data from lower layers. Since the investigator knows a priori the relationship between adjacent layers via the input/output relationship, he can evaluate the actual data of a layer against the data that should exist.

Knowing how to identify intentionally modified layers is not a simple task that can be done quickly and easily. In theory, the difference between layers is easy to detect; in practice, however, the investigator's lack of resources – time and money – presents difficulties. Specially designed tools can be used to quickly extract the pertinent information from each layer. A wrapper tool can then take that information, apply the input/output relationship formulas, and return a percentage that indicates the likelihood of a layer having been intentionally altered, thereby providing the investigator with the information necessary to determine whether a layer has in fact been altered.

The results of several test cases of the cross-validation program are promising. While the tools are not precise enough to detect all anomalous layers without producing false positives, they do produce results accurate enough to yield a range of percentages that can be analyzed further by the investigator. Although the tool does not fully automate the analysis for the investigator, it does mitigate the audit reduction problem with respect to file system layers.

This paper introduces the concept of cross-validation for file system images. First, the basics of mathematical cross-validation are discussed. Next, the technique for cross-validating file systems is covered, including the major differences between mathematical sets and file system images. The technique is then illustrated through an example using the FAT12 file system. Finally, the viability of cross-validation for other file systems is discussed.

A.3 Cross-Validation

Cross-validation (CV) is the technique by which known information is tested against a set of unknown information; the result is an error ratio. First, a subset of the data, called the training set, is acquired. That data is used to develop a model of what the unknown information should be. The unknown information, or test set, is then juxtaposed against the newly derived model. The result of the juxtaposition is an error ratio that shows how closely the model fits the test set.

A mathematical example of cross-validation is shown in Figure 1:

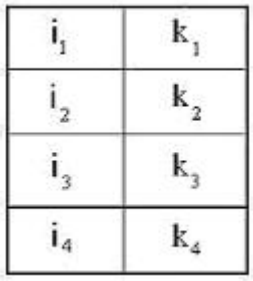

Figure A.1: Mathematical cross-validation.

The first column contains four elements, out of which the first three - i1, i2, i3 - comprise the training set. A model, i4‘, is derived either based on a pattern from those three elements or some known rules that govern how the set is constructed. Next, i4 is juxtaposed against i4‘. The difference(s) between them is the error ratio. The same is done for the second column. Finally, the two error ratios are averaged to arrive at the final error ratio.

Numerous applications of cross-validation exist, including the evaluation of AI agents. The typical data set for cross-validation usually exceeds one hundred items, a computationally expensive number. However, the resulting error ratio is a strong and accurate indicator of the model’s prediction. The accuracy obtained by cross-validation is what makes it so appealing. The computational expenses are fully justified by the level of accuracy that is obtained.

Cross-validation is effective for learning if information is structured as it should be. In mathematics, when a formula is applied to an input, there can be zero, one, or multiple

results. When a formula is one-to-one, however, there is always one output for every input. Cross-validation is effective in cases where a data set whose structure is determined by some one-to-one mapping between the elements in the set exists [28]. One can use a subset of that data and effectively predict the other subset. A comparison of the prediction to the actual subset shows how closely the sets conform to the rules or patterns that govern them.

A.3.1 Cross-Validation and File Systems

File system layers contain enough similarities to mathematical sets to allow for the use of cross-validation. Cross-validation begins with a set of data, splits the set into subsets, and then makes a prediction about the information contained in one subset based on the other subset. This occurs in file system layers. The overall set can be thought of as a file system image. Instead of subsets, file systems have abstract layers. Each lower layer is used to form higher layers according to the rules of the file system. If an investigator has full access to the information at the lower layers, he can predict what information the next higher layer should contain. This is essentially the process of cross-validation: employing known information to form a model and then comparing that model to the unknown information.

Another similarity between mathematical sets and file system layers is the parallel between one-to-one functions and the input-output relationship. The latter – the rule that transforms a lower-level layer into a higher-level layer, and vice versa – is essentially the same concept as a mathematical function. Two types of layers can affect the input-output relationship: lossy and lossless [6]. A lossy layer is an abstract layer that can contain errors when produced by the input-output relationship. The lossy layer is designed such that information can be lost or misinterpreted in the process of transforming a lower layer into the lossy layer. Typically, lossy layers reside at the application level. For example, source code written in C++ may be lossy if it is compiled differently by various compilers.

Lossless layers, on the other hand, do not introduce the possibility of data being lost or misinterpreted when formed from lower layers. One can think of lossless layers as either a one-to-one mapping or a many-to-one mapping, depending on the type of input and output. In both cases, the output is consistent, and therefore no information is lost. File system layers are considered lossless. Recall that a one-to-one mapping for cross-validating mathematical sets allows for accurate models of test sets to be produced; the same holds true for file system layers. Since the input and the input-output relationship are known,

and the file system layers are lossless, the model of the test set will produce accurate results if the test set has not been altered.

The differences between file system images and mathematical sets require that a modified version of cross-validation be used for the former. When mathematicians cross-validate, they do multiple iterations over a data set. A data set of 500 items, for example, may be broken into ten subsets. Nine of the ten subsets are used during each iteration, for a total of ten iterations. This method of validating file system layers is not viable; the amount of information contained in a file system image is far less than one hundred data units, and many of the file system datum are only used in two or three of the layers. Thus, the cross-validation used for file systems is not a "true" form of CV. This form of CV must measure the discrepancies between layers that share information, but it cannot perform the multiple iterations done in mathematics.

The CV formula is derived by a process of examining how each layer should be formed and summing those values. The first layer – the raw system image – is a boundary layer that is a string of bits and not actually an abstract layer, so it is not included in this formula. Each of the remaining layers is used. Since every layer contains different types and amounts of information, the formula should be weighted. Some layers carry more useful information than others and thus carry more weight. Exactly how much weight a layer should carry is left to the investigator to determine; this requires a combination of knowledge of the file system type and common hacker tricks. An example formula using the FAT12 file system is shown in the following section.

An accurate means for determining each layer's CV value is difficult to establish. The assignment of weights to each layer requires from the investigator both a knowledge of the layer's importance and an understanding of the minute details. There exists information extraneous to CV, alongside important information. Thus, an ad hoc approach must be taken when deriving a layer's formula.

A.4 FAT12 Example

The following is a brief description of the FAT12 file system's abstract layers. For more complete discussions, see [7] and [10].

A.4.1 FAT12 Overview

The FAT12 file system consists of four areas: the Boot Sector, the Data Area, the root directory and the File Allocation Table (FAT). The Boot Sector contains all of the information needed for the OS to call disk driver functions to access sectors. It contains the sizes of the various structures for that particular file system image, e.g. the number of FATs. Each sector, called a cluster, is located in the Data Area. This area comprises the contents of every file and directory. The FAT is a linked list structure that stores information about each cluster. It specifies where the next cluster is and if a cluster is unallocated, reserved, or bad. The root directory is the topmost directory, typically c: for hard drives or a: for floppy disks. While all subdirectories in FAT12 are stored as files, the root directory is not; it is its own entity.

Table 1 shows the seven abstract layers for the FAT file system.

Layer	Input	Output
1	Raw file system image	Boot Sector values
2	File system image and Boot Sector values	FAT and Data Areas
3	FAT Area, FAT Entry size	FAT Entries
4	Data Area and Cluster size	Clusters
5	Raw cluster content and content type	Formatted cluster content
6	Starting cluster and FAT Entries	Linked list of clusters
7	List of clusters, clusters, formatted cluster content and type	All Directory Entries in a directory or raw content of a file

Figure A.2: The FAT12 file system layers.

To retrieve the directory entries or the raw contents of files, the raw file system image is obtained (layer 1), and the boot sector values are extracted. The FAT and Data Area is then extracted using the file system image and the boot sector values (layer 2). Next, the FAT Area and FAT entry size are used to obtain the FAT entries (layer 3). The

fourth layer uses the Data Area and Cluster Size information to retrieve the clusters. Next, the raw cluster content from the fourth layer along with the content type of those clusters is used to retrieve the formatted cluster content. If the cluster content forms a file, then the extraction process stops here at the fifth layer. Otherwise, if the content is a subdirectory, the starting cluster and the FAT entries from layer three are used to gather the list of directory clusters. Finally, the list of clusters, the clusters themselves, and the formatted cluster content along with their type produce all of the directory entries in a directory or the raw contents of a file.

A.4.2 Cross-Validating FAT12

All of the layers in FAT12 (except for the raw system image) contain valuable information, so each must be made part of the CV formula. The layers should be analyzed in turn by asking, "What could be erroneous or falsified at this layer?" The answer to this question provides the elements of the formula. For example, what could be erroneous at layer 6 in FAT12? Perhaps the starting cluster could be modified or the FAT entries for a file might be incomplete.

The following is a high-level discussion of each layer's importance in the FAT12 CV formula. When constructing this formula, the assumption is that the file system appears normal. Since a "wiped" file system shows obvious signs of having been altered, using CV to approximate the chances of deception would be pointless.

Layer 1: Little risk of deception exists at this layer. One should make sure that the boot sector values are valid. Several items to validate: the number of FATs is nonzero, the logical sector size is a power of two and a value of at least 512, and the number of root directory entries is a multiple of the number of directory entries per logical sector.
Layer 2: Here exists a slightly higher possibility of deception. For example, check that the FAT is sufficiently large so that it contains the bit for the last cluster. Otherwise the last cluster of the Data Area will be inaccessible.
Layer 3: This is another low-risk layer. Both the likelihood and effect of erroneous values here are minimal. One threat is that of an incorrect FAT entry size, which would lead to an incomplete retrieval of the FAT entries.

Layer 4: Once again, there is no substantial threat for deception at this layer. If the Data Area was successfully extracted (Layer 2) and the cluster size is correct (Layer 1), the clusters will be retrieved as expected. However, if there was a problem at Layer 1 or 2, then this layer will not be correct. [Herein lies the strength of CV: even if incorrect values are returned at one layer, checking those values at other layers will show the discrepancy.]
Layer 5: This layer is critical and should be weighted as such. If a cluster's content type is a file, this layer is its boundary layer. That is, the file system will go through no further steps since all of the file's information is now known. If, however, a cluster's content type is a Directory Entry, then this layer is not a boundary layer. Thus, the correct content type is essential for returning the correctly formatted cluster content.
Layer 6: Data is most likely to be hidden at this layer. Both the starting cluster and FAT Entries can easily be altered to point to incorrect clusters, thereby causing an incorrect list of clusters to be returned.
Layer 7: The final layer is the most important layer. Each of the seven layers adds information, culminating in the output of Layer 7. The most weight should therefore be given to this layer. Several heuristics should be employed to ensure the validity of the list of clusters, e.g. checking the cluster content type or matching the number of FAT entries to the file size listed in a file's Directory Entry. Table 2 lists these seven layers, their relative importance to the integrity of the file system and risk of being altered, along with their respective weights.

Table 2 The CV formula assigns an overall weight to each layer, and each layer assigns a weight to each piece of information contained therein. For FAT12, Layer 7 should be given the greatest weight, and Layer 1 should be given the smallest weight.

Within each layer, the individual pieces of information must be weighted according to the information's impact on data integrity and the likelihood of tampering. In Layer 6, for example, the most importance should be placed on the completeness of the linked list of clusters. If the list is too small, there is a strong possibility of tampering. The final formula reached for a FAT12 file system image is:

$$Er = 0.05(L1) + 0.1(L2) + 0.05(L3) + 0.1(L4) + 0.2(L5) + 0.2(L6) + 0.3(L7)$$

Layer	Relative Importance/Risk	Weight
1	Low / Low	0.05
2	Low / Moderate	0.1
3	Low / Low	0.05
4	Moderate / Low	0.1
5	High / Moderate	0.2
6	Moderate / High	0.2
7	High / High	0.3

Figure A.3: Weights for file system layers.

A.4.3 Implementation

Performing CV requires access to each layer. For this research, "Sleuth Kit" is used to retrieve each layer. A set of tools written for Unix-based systems, the Sleuth Kit has evolved from Dan Farmer and Wietse Venema's "The Coroner's Toolkit." The Sleuth Kit is designed to gather file system information based on the four file system layers: the file system layer, content layer, meta data layer and name layer. The seven FAT12 layers are accessed by employing the Sleuth Kit with the addition of a Perl wrapper script. The wrapper first runs the Sleuth Kit to retrieve the Boot Sector values (Layer 1), then stores that information and repeatedly calls the Sleuth Kit tool responsible for each subsequent layer until all of the necessary information has been gathered (Layers 2-7). Next, the wrapper uses the information from Layer 1 to make a prediction about the information in Layer 2. The model of Layer 2 is then juxtaposed against the Layer 2 values collected by the Sleuth Kit; any discrepancies between the two are stored, and an error value is assigned to that layer. The process of using the actual values from the last tested layer to form a model, and then testing that model against the next layer, is continued through Layer 7. Finally, the error ratio is derived by weighting and summing the stored values for each layer. 3.4

A.4.4 Results

The overall results of cross-validating correct FAT12 images are promising. Generally, the results are consistent and intuitive. The error ratio falls in the 90-100% range for file system images that contain no forms of deception. Only when an image has been extensively used does the error ratio drop to 90%. An image that is highly fragmented and contains deleted files in directory entries will thus appear to have traces of deception. The cause of this is a heuristic that regards those items as suspicious. These are the only known false-negatives produced in the cross-validation process. Nevertheless, an error ratio of 95% is sufficiently high for the image not to be investigated more closely.

Cross-validating highly suspect FAT12 images also produces promising results. The error ratio for highly suspect images – ones that contain a small number of files and one or more major forms of tampering, or many files and multiple forms of tampering – is below 80%. If an investigator receives an error ratio well below 80% after running the CV against a FAT12 image, he knows that the image has almost certainly been tampered with and should be more closely investigated.

The main difficulty with cross-validating FAT12 images is false-positive error ratios. A file system image that contains one major form of deception - perhaps a file size/FAT entries mismatch - but is otherwise pristine is difficult to analyze. The resulting error ratio will not properly reflect the deception because it will still be rather high. The typical inconclusive false-positive error ratio is between 80 and 95%. The investigator is then left with the decision as to whether further investigation is merited.

Handling false-positive error ratios demonstrates the importance of human analysis of CV results. The dilemma facing CV is that on one hand, cross-validation should catch as many forms of deception as possible; the investigator should be alerted to all forms of tampering. But on the other hand, the point of cross-validating is to save time. If the investigator is forced to analyze every false-negative, he will gain little, if anything, from the process. An investigator must thus make use of the results while remaining critical of them.

FAT12 is not the best file system for cross-validation. Since FAT12 file system images are at most 8MB, there is little need for a semi-automated means of analyzing them (unless an investigator has to analyze dozens of floppies). The small size of FAT12 images limits the granularity of the cross-validation. Remember that most mathematical usages

of CV involve data sets in the hundreds. To use a CV formula that acts on such a small amount of data is naturally going to result in a lack of precision.

The lack of absolute precision should not discourage the use of CV for FAT12; cross-validation is still effective for this use. As shown above, the majority of FAT12 images do result in accurate error ratios. Investigators can use CV to eliminate a number of unmodified images from a set of FAT12 images that need to be investigated. In the case of borderline error ratios (80-95%), the investigator can make a decision as to whether he will analyze them based on the amount of resources available and his intuition on the matter.

The use of FAT12 also demonstrates the basic premise of file system cross-validation. Every step – producing a FAT12 CV formula, gathering layer evidence and applying the formula to the layers – can be utilized for other file systems. Naturally the CV formulas for FAT32 and NTFS, with FAT12 as their legacy, can be developed based on FAT12's formula. But even more robust file systems, such as Red Hat's ext3, are well suited for being cross-validated. The important point is that cross-validation can be effectively applied to file systems, even to those of the coarsest granularity.

A.5 Future Research

This paper lays the groundwork for further research into the cross-validation of file system layers. The need for validating evidence in a semi-automated manner will increase as file systems grow in storage capacity and forensic investigations become more rigorous. The next step is to develop effective cross-validation methods for modern file systems, such as NTFS and ext3fs. These more robust, sophisticated file systems offer more information and finer granularity for cross-validation. Also, as the amount of storage increases, so too does the need for a semi-automated means of validation.

Another enhancement that could greatly improve results is the inclusion of lower layer evidence, such as the partition table, along with application layer cross-validation. Evidence at the file system layers is quite useful, but should be supplemented with information about the disk layout and partitions. This low-level information would help corroborate evidence at the lower layers of the file system level. Cross-validating application layer evidence is probably the most useful addition to file system cross-validation. With threats of stenography, file splitting, and encryption, a wealth of data could be cross-validated. But

most importantly, combining the cross-validation of file system layers with application layers would help to uncover evidence that could not easily be discovered by focusing exclusively on one at a time. For instance, if a certain file type is known to have a data descriptor string in one of its data clusters, that string can be cross-validated against the actual contents of the file. This feature becomes more attractive when one considers that file types may have multiple unique characteristics that can be cross-validated, and that this process can be automated over the entire file system.

The ultimate goal in file system cross-validation is the discovery of an effective means for cross-validating storage media that contain multiple partitions of different file system types and virtual file systems. Many sophisticated computer users now use machines that can be booted into several different operating systems, and the number of people using virtual machines like VMWare is rapidly increasing. Combing through all of the evidence located on multiple partitions is tedious and often difficult. Cross-validating the entire media would provide a useful forensic snapshot, allowing the investigator to quickly discover leads. The error ratio produced, along with the output of the discrepancies, would serve as a form of evidence for the investigator. Such evidence is especially important when dealing with complex evidence media, for the likelihood of an investigator making an error is greater.

A.6 Conclusion

This paper has focused on the development of the basic concepts necessary for cross-validation of file system layers for computer forensics. Cross-validation has a well-established history of effectively evaluating the performance and predictability of data sets in mathematics and certain fields of computer science. Computer forensics strives for precision and accuracy, and so the adoption of cross-validation for this field is a natural one.

A recurring theme in this paper is the need for human analysis of the cross-validation; the error ratio alone is not sufficient for uncovering all cases of hidden or falsified data. As is seen in cases of a moderate error ratio (80-95%), an investigator cannot quickly gather what the results mean. Usually, the investigator must check the output of the CV to see what discrepancies occurred so that he can make sense of the error ratio. If he were to go by the error ratio alone, he would certainly either waste time investigating cases unnecessarily, or he would ignore cases that should have been further investigated.

Ideally, cross-validation of file system layers would allow an investigator to run the CV script and receive an authoritative "investigate" or "do not investigate." However, such a Turing machine-like program is not possible; mathematical certainty of a compromised file system would presuppose the investigator's knowing all possible forms of compromising. Cross-validation instead provides a quick and effective means for understanding how suspicious a file system is. CV can also help an investigator be more thorough in his investigations by uncovering evidence that he may have overlooked.

The benefits of cross-validation far exceed the initial difficulties that will be faced when developing these methods. A forensics investigator's resources are valuable and often in short supply. Every tool or technique that can assist him should be explored, and cross-validation is such a technique. While the method is not without difficulties, its overall precision and ability to save time, money, and resources make it an invaluable aid to the forensic investigator.