Computer Forensics *In Forensis*[*]

Sean Peisert[†] and Matt Bishop[‡]
Department of Computer Science
University of California, Davis
{peisert,bishop}@cs.ucdavis.edu

Keith Marzullo[§]
Dept. of Computer Science & Engineering
University of California, San Diego
marzullo@cs.ucsd.edu

Abstract

Different users apply computer forensic systems, models, and terminology in very different ways. They often make incompatible assumptions and reach different conclusions about the validity and accuracy of the methods they use to log, audit, and present forensic data. This is problematic, because these fields are related, and results from one can be meaningful to the others. We present several forensic systems and discuss situations in which they produce valid and accurate conclusions and also situations in which their accuracy is suspect. We also present forensic models and discuss areas in which they are useful and areas in which they could be augmented. Finally, we present some recommendations about how computer scientists, forensic practitioners, lawyers, and judges could build more complete models of forensics that take into account appropriate legal details and lead to scientifically valid forensic analysis.

1: Introduction

> "The principle of science, the definition, almost, is the following: *The test of all knowledge is experiment.* Experiment is the *sole judge* of scientific "truth."
> —Nobel Laureate Richard P. Feynman,
> California Institute of Technology, Sept. 26, 1961. [21]

Who attacked this computer system? What actions did they take? What damage did they do? With what degree of certainty, and under what assumptions, do we make these assertions? Will these assertions be acceptable in a court? These questions are asked during the computer forensic analysis process. They are often hard to answer in practice. Computer scientists and forensic practitioners have both made headway on developing

[*] This paper is an expanded version of an earlier paper published with the same title in *ACM Operating Systems Review (OSR)*, **42**(2), Apr. 2008 [48].

[†] This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

[‡]Matt Bishop gratefully acknowledges the support of award number CNS-0716827 from the National Science Foundation to the University of California at Davis.

[§]Keith Marzullo did this work in part under the support of the National Science Foundation under Grant Number CNS-0546686.

functional systems for forensic analysis. Some of those systems are based on theoretical models that help to construct complete solutions, but there are serious and important gaps in these systems.

The field of *computer forensics* has become a critical part of legal systems throughout the world. As early as 2002 the FBI stated that "fifty percent of the cases the FBI now opens involve a computer" [26]. However, the accuracy of the methods—and therefore the extent to which forensic data should be admissible—is not yet well understood. Therefore, we are not yet able to make the kinds of claims about computer forensics that can be made about other kinds of forensic evidence that has been studied more completely, such as DNA analysis. The accuracy of DNA analysis is well understood by experts, and the results have been transformational both in current and previous court cases. DNA evidence has been instrumental in convicting criminals, and clearing people who have been wrongly convicted and imprisoned. DNA evidence condenses to a single number (alleles) with a very small, and well defined, probability of error. On the other hand, computer forensic evidence has matured without foundational research to identify broad scientific standards, and without underlying science to support its use as evidence. Another key difference between DNA and computer forensic data is that DNA evidence takes the form of tangible physical "objects" created by physical events. Contrast these to *computer objects* that are created in a *virtual world* by *computer events*.

Computer-based evidence has only recently become common in court proceedings, but its impact in the legal system has been significant. Cases are frequently decided on evidence obtained from computer systems—evidence that many experts claim is unreliable. Consider the recent case *State of Connecticut v. Julie Amero* in Norwich, Connecticut [16]. An elementary school substitute teacher, Ms. Amero was accused, tried, and convicted of contributing to the delinquency of minors because a spyware-infected school computer in her class displayed pornographic sites' pop-ups during her lecture. The legal system's lack of technical awareness resulted in a conviction that, while eventually overturned, permanently impacted Ms. Amero's life and diminished the credibility of our legal system. Judges and juries make inappropriate assumptions because they expect that computer forensic evidence in real life is as reliable and conclusive as it is on television. The impact of these assumptions cannot be undone merely by reversing a court decision. In many cases such as these, the forensic tools being used are accurate, but the *assumptions* made about them are wrong.

Most judges and lawyers do not understand actions and objects inside computer systems well. Therefore, the legal system is often in the dark as to the validity, or even the significance, of computer evidence. In many ways, computer forensics is behind other methods such as fingerprint analysis, trace evidence of soil samples, cigar ash, the timing of insect infesting corpses, and the chemical traces of poisoning [62] because there have been fewer efforts to measure and improve its accuracy.

For example, one problem arises when the traces of an attack have been altered so that the attack is hidden [60]. In this case, the data itself can be inaccurate or misleading. In other cases, the data may be accurate but not support the conclusions that are drawn. As an example, Mary may own a file, but there is no way to show that Tom was logged in to Mary's account during the time period in question.

Many technical disciplines used in forensic testimony produce results with well-defined margins of error. When technical evidence is presented, an expert witness is frequently asked to answer specific questions ("How fast would the car have had to be going for the metal to have crumpled like this?"). But in computer forensics, analysts are asked to tell

complete stories—the meaning of a series of events, how those events were triggered, and who triggered them. Unfortunately, an expert may not be able to justify their answer rigorously because the limits of the methods used in computer forensics are not understood as well as those in, say, DNA analysis.

Few analysts are currently challenged to defend the validity of the results that their tools present. One reason for this could be that some analysts may feel that they can claim that the forensic software that they use has been certified by the U.S. National Institute of Standards and Technology (NIST) [39]. However, NIST tests how well tools conform to specific requirements of law enforcement staff; that is, against what the forensic tools are supposed to do. For example, according to the NIST Deleted File Recovery specification [40], the testing assumes that "the deleted file recovery tools are used in a forensically sound environment" (p. 6). In order to evaluate a particular instance of use of the tool, the analyst must know the characteristics of the environment in which the tool works well, and where it works poorly. Further, the NIST program does not provide metrics to determine how accurately a tool works; it simply determines whether a set of requirements are, or are not, met. In practice, "shades of grey" complement the NIST work on determining where the lines of "black" and "white" lie.

Computer scientists can take steps to move computer forensics into a more rigorous position as a science by being able to make well-reasoned and concrete claims about the accuracy and validity of conclusions presented in court. Our goal is to try to point out the confusion between forensic practitioners, law enforcement officials, and computer scientists, and to encourage a dialog, in hopes that the groups will begin to work more closely together in order to solve the critical problems that exist in the application of computer science to legal issues. We seek to help the different groups understand the steps that must be taken in order to make claims about computer forensic data, and under what conditions those claims are appropriate and when they are not. In this paper, Section 2 discusses the varying terminology used by the different people involved with computer forensics. Section 3 discusses the importance of developing common understanding among researchers and practitioners who use forensic techniques *outside* of the courtroom. Section 4 discusses the technology used by forensic practitioners, and that which has been developed by computer scientists. Section 5 discusses the notion of forensic models, how different groups use the term, and how the concept can be unified. Section 6 presents two case studies where forensic analysis is hindered by the lack of a unified model. Section 7 presents our conclusions on how forensic systems can ultimately be improved to advance computer forensics as a science.

2: Forensic Language and Terminology

Those involved in computer forensics often do not understand one other. Groups have evolved separately with only little interaction. Each group has largely separate conferences, journals, and research locations, and few attempts have successfully brought these groups together. Indeed, the language used to describe computer forensics—and even the definition of the term itself—varies considerably among those who study and practice it: computer scientists, commercial ventures, practitioners, and the legal profession. As a result, it is difficult for these groups to communicate and understand each others' goals.

Legal specialists commonly refer only to the analysis, rather than the collection, of enhanced data: "The tools and techniques to recover, preserve, and examine data stored or

transmitted in binary form." [28] By way of contrast, computer scientists have defined it as "[v]alid tools and techniques applied against computer networks, systems, peripherals, software, data, and/or users – to identify actors, actions, and/or states of interest." [66]

Even within the computer science discipline, there is disagreement about terminology. "Software forensics" has been defined as "tracing code to its authors." [56] Some computer scientists focus largely on the *examination* of filesystem data [13], whereas others also include the *collection* of data [11, 15, 20, 34, 35, 57].

The term *forensics* derives from the Latin *forensis*, which meant "in open court or public," which itself comes from the term *forum*, referring to an actual location—a "public square or marketplace used for judicial and other business." [1] Contemporary use of the word *forensics*, therefore, generally continues to relate to law, and has come to mean "scientific tests or techniques used with the detection of crime." Thus, *computer forensics* implies a connection between computers, the scientific method, and crime detection. *Digital forensics* is largely used interchangeably with *computer forensics*, but implies the inclusion of devices other than general-purpose computer systems, such as network devices, cell phones, and other devices with embedded systems. However, largely everyone *except* academic computer science researchers use the term in connection with the law. Many computer scientists have simply been using the word "forensics" as "a process of logging, collecting, and auditing or analyzing data in a *post hoc* investigation."

The result of a lack of common language has been that frequently, the groups do not understand what each other considers important. Most computer forensic solutions in common use by law enforcement have not advanced significantly since *The Coroner's Toolkit (TCT)* ,which was developed in 1999. To be sure, we have had significant technological progress, and tools like *Sleuth Kit*, that examine the filesystem, have been large steps forward. In addition, our ability to gather data has improved greatly; we now even have the ability to determine the contents of data stored in semiconductors [25]. However, there is still little understanding in any community as to *when* and *how* such techniques and tools are applicable in a court of law, and to what extent claims can be made about the data derived from them. Dan Farmer and Wietse Venema noted:

> "Certainly the current set of software tools is not terribly compelling. Our own Coroner's Toolkit, while at times useful, could be much improved upon. Other packages—most notably the Sleuth Kit and EnCase—are worthy efforts, but they still have far to go. It's too bad that we have not progressed much further than the erstwhile `dd` copying program, but automated capture and analysis are very difficult." [20]

One of the reasons for the difference in terminology is the difference in goals. To computer scientists, computer audit trails have other uses than computer forensic data. For example, the analysis of audit trails can provide assurance that a machine is operating according to functional, reliability, and performance specifications. Audit trails may be used for billing and accounting purposes. The needs of accounting and debugging are often quite different from forensics.

Computer audit trails are not the only type of *computer forensic data* that law enforcement uses. For example, at present, the vast majority (80%) of cases considered by judges and law enforcement to be "computer crime" involve child pornography [36], and therefore, the vast majority of the forensic data used are simply files on a disk (or possibly files that have been deleted from the file catalog and subsequently recovered by analysts).

Much more "computer crime" exists than law enforcement acknowledges or identifies, and there are many techniques that law enforcement is largely unaware of. Because the focus of law enforcement is on recovering files rather than discovering how the files entered the system, there is little emphasis on enhancing systems to collect such data. Therefore, the vastly enhanced solutions that computer scientists offer forensic practitioners are seen as unnecessary by the practitioners.

Computer scientists who are working with law enforcement officials should be driven by legal goals, but they need to understand those goals. Computer scientists also need to make their *capabilities* known. Forensic practitioners need to establish and communicate what they are looking for. Often, that is to tie an action or object to a specific person. Just as often, computer science cannot establish that fact with any reasonable degree of certainty. So, the question for judges and lawmakers is how precise the evidence needs to be. Must the possible range of perpetrators be reduced to one in ten or one in 1000? It is up to practitioners and policy makers, with input from computer scientists, to determine a set of requirements that can be implemented. It is up to computer scientists to build systems that fit these requirements, to validate and verify that the systems meet the requirements, and to specify the conditions under which the systems meet the requirements. Similar techniques can be used to provide defensible solutions not just to forensic practitioners and lawyers, but to any group (such as states and counties purchasing electronic voting machines or hospitals purchasing highly sensitive medical equipment) that need systems with measurable and verifiable accuracy.

3: Uses of Forensic Techniques Outside of the Courtroom

Forensic audit trails can have many uses. The courtroom is one of them, but not the only one. The notion that computer scientists often ignore legal issues has a reason: sometimes one simply does want to know what happened previously, for example, for the purpose of fixing an error, debugging, performance measurement, or compliance verification (e.g., HIPAA or Sarbanes-Oxley requirements, in the United States). Large institutions, be they a military sites, large companies, or government agencies, frequently simply wish to understand the events happening within their networks.

At other times, the issue could be specific and critical: to determine the outcome of a national election. Each entity needs to take into account different things than legal investigations, and certainly different things (such as a network device or a closed-circuit video) than one would need to take into account to investigate a single host. However, those needs are not yet well defined, and neither theoretical models nor implementations take them into account. For example, in some cases, the needs may even directly contradict, such as the ability to reverse engineer a malfunctioning voting machine vs. anonymity of the voter.

Many jurisdictions in the United States now rely on electronic voting machines (DREs or optical scanning systems) to determine the winners of elections. Electronic voting machines are a good example of a class of machines that produce output which should be beyond question. The public assumes that the ballots cast are recorded accurately, precisely, and reliably. The public assumes that the machines allow the correct number of votes per race per valid, registered voter, while protecting both anonymity and secrecy of the ballot. They also assume that the machines enforce other desired qualities, such as allowing each

authorized voter to cast no more than one ballot, and preventing ballots from being tied to the voter who cast it. But many engineering studies have challenged these assumptions [8, 9, 65, 22, 33]. Worse, assumptions differ between jurisdictions. For example, election officials in some states can have a precinct "revoted" (by recalling voters in that precinct) should they find problems with the management of the election there. However, other states (notably California) can invalidate elections, but cannot recall a precinct's voters to revote.

Those studies of electronic voting machines are important because many of the requirements of electronic voting systems also apply to tools for computer forensics on general-purpose computer systems. And, indeed, if electronic voting systems were redesigned with forensic issues in mind, then forensic tools and techniques could help to measure the accuracy and reliability of electronic voting machines. The problem is that neither reliability and accuracy, nor the steps that must be taken to ensure them, are clear.

Consider the the electronic voting machines used in Goshen, New York, which functioned perfectly well until the vote tally exceeded 999 votes. At that point, the counter reset to zero [2]. The results of the election were invalid, and true counts will never be known. Here, the (implicit) assumption was that no voting machine would need to record more than 999 votes for a candidate. In the environment in which those systems were used, the assumption failed to match reality.

In many jurisdictions, electronic voting systems (DREs) are required to print a paper record of the votes cast, display it to the voter, and store this representation as well as the electronic representation of the vote. The paper is called a "voter verified paper audit trail" (VVPAT) and is used during election audits. If voters consistently cooperate, it works well for verifying that the system recorded votes properly, because each electronic ballot can be compared to the paper ballot. However, it works poorly as a forensic audit mechanism, because the paper ballots are not tied to specific events within the DRE. If there is an error, the VVPAT cannot provide conclusive evidence of what happened. Thus, asserting that a VVPAT is an "audit trail" for DREs is incorrect. It is an audit trail for votes, not for the systems that record the votes [64].

We now rely on electronic voting machines to provide the evidence which determines the results of our democracy in the United States. Yet two things are clear. First, is possible to make a system that performs its assigned tasks reliably. Given its narrow set of operating parameters, a voting system need not be a hugely complex system to design and implement. Second, however, is that no system is likely to ever stand on its own. It is always subject to the human policies which guide its use. We can live with a system that is confidential if the voter stands behind a curtain, as long as we know in advance that a curtain of a particular design is required. But what really are the assumptions that are made when it comes to electronic voting? The voting machine vendors certainly have not published a comprehensive list. The reality is that we simply do not have a thorough knowledge of the situation because metrics for scientifically measuring the accuracy, reliability, and validity of the machines under varying procedural assumptions, have not been developed.

4: Forensic Systems

In practice, forensic analysis of a computer system involves identifying suspicious objects or events and then examining them in enough detail to form a hypothesis as to their cause and effect. Data for forensic analysis can be collected by introspection of a virtual

machine during deterministic replay [17], as long as nondeterministic events can be logged, the overhead is acceptable, and the target machine has only a single processor (because multiprocessors introduce nondeterminism). Specialized hardware [38] can make nondeterministic event logging practical, but this kind of hardware is rarely available. Most existing tools simply operate on a live, running system, and look both at system and network-level events and files on a disk.

The concepts of "logging" and "auditing" have been around for a long time. Anderson and Bonyun first proposed use of audit trails on computer systems [4, 10]. They discussed the merits of certain data and the placement of mechanisms to capture that data, but did not discuss how the process of selecting data could be generalized. Throughout the early evolution of audit trails, sophisticated logging capabilities were developed for multiple platforms. However, the purpose was purely an ad hoc method of capturing data thought to be useful for investigatory purposes, and was not intended for legal use.

Two approaches to auditing are *state-based* and *transition-based* [7]. State-based auditing periodically samples the state of the system. This approach is imprecise—important information may be missed—and risks slowing a system unacceptably. Transition-based auditing monitors for specific events, which are more easily recorded. Both state-based and transition-based logging require deciding in advance on levels of granularity to record. But state-based logging also requires deciding the frequency with which to save state information, and finding the right balance between generated load and missing key information is difficult. As an example, debugging breakpoints and fault tolerance checkpoints are instances of state-based logging mechanisms. Both of these tasks can involve an iterative processes of logging, analysis, and replay. With debugging, when a bug is suspected, the analyst might insert a series of breakpoints to check the values of a number of variables at various points in time. If the analyst determines she needs more information, or if a fault occurs again before the breakpoint is triggered, then the breakpoints can be changed and the program re-run. But security and forensics do not allow an exact replay of the series of events leading to a suspected attack unless deterministic replay is used. The relevant information is only seen once. If an attack occurs between two captures of state information, traces of the attack may already have been removed by the time the second snapshot is taken. Therefore, in forensics, one must rely on specific events to trigger the logging mechanism. Thus, transition-based logging is generally the most appropriate.

Examples of successful tools include *Tripwire* [29], which records information about files, and *TCP Wrappers* [61], which records information about network communications. Additionally, *The Coroner's Toolkit*,[1] *Sleuth Kit*,[2] *EnCase*,[3] and the *Forensic Tool Kit (FTK)*,[4] are all useful for analyzing filesystems and files that are present or have been recently deleted from filesystems.

Today, UNIX system log (syslog) entries, and the equivalents on other operating systems, are commonly used forensic data sources. However, these mechanisms were designed for debugging purposes for programmers and system administrators, and not for forensics [3]. Similarly, the Sun Basic Security Module (BSM) [41] and cross-platform successors are constructed based on high-level assumptions about what events are important to security, and not to answer specific forensic questions such as who committed a certain action. The

[1] http://www.porcupine.org/forensics/tct.html
[2] http://www.sleuthkit.org/sleuthkit/
[3] http://www.guidancesoftware.com/
[4] http://www.accessdata.com/common/pagedetail.aspx?PageCode=homepage

most successful forensic work has involved unifying these tools using a "toolbox" approach [20, 42] that combines application-level mechanisms with low-level memory inspection and other state-based analysis techniques.

Most of the techniques that are currently used to gather information in court are an essential part of the forensic process, and probably should continue to be used. But we assert that their application is flawed. None of the forensic techniques currently used in court are sufficient to justify claims that implicate a specific person. It is not enough to recover a deleted file or view a standard system log. One has to know the history of files and the events that led up to their creation, viewing, deletion, and modification. Consider again the Amero case mentioned in Section 1. A criminal conviction requires (among other things) proving beyond a reasonable doubt that Ms. Amero intentionally downloaded child pornography onto the school's computer. Images might appear on a disk without the computer user knowing about them for many reasons—pop-up images on web sites may download files in the background and save them in the cache; the images could be part of unsolicited "spam" email; another person may simply have downloaded them, either to view the pornography themselves, or to implicate someone else. Many forms of malware are capable of commandeering a computer in order to store and/or redistribute porn. Such malware would have explained the images as well as the corresponding changes to the browser's history file, all done without Ms. Amero's consent or knowledge.

However, the forensic software used in the vast majority of court cases cannot make the distinction among these methods of file creation. Such software does not provide sufficient information to enable an analyst to reconstruct previous events rather than just objects, particularly when those events appear "ordinary," such as when committed by insiders [45]. In court, a jury must consider questions that are not as straightforward as whether a file exists or an action has taken place. The jury needs to know *how* the file got there and *who* took the action.

Sometimes information is accurate, but the claims derived from it are not supported by scientific experiments. More often, we are uncertain of the accuracy of the data. The ability to combine the information from forensic mechanisms to arrive at an understanding of the events behind an attack often relies on luck rather than on methodical planning. Specifically, an analyst who finds the machine immediately after an intrusion has taken place can gather information about that system's state before the relevant components are overwritten or altered. Otherwise, the analyst is forced to reconstruct the state from incomplete logs and the current, different state—and as the information needed to do the reconstruction is rarely available, often the analyst must guess. The quality of the reconstruction depends on the quality of the evidence present, and the ability of the analyst to deduce changes and events from that information. Often, tools cannot provide sufficient information because the level of granularity of the information they gather is far too high, or the data is difficult to correlate, or, even when several tools are used together, they do not record information from enough sources to perform a thorough forensic analysis. We are left with the question of which existing techniques work at all, and if so, when, and to do what?

Forensic analysis requires better data, and in particular data that can be collected without having to predict specific security vulnerabilities or exploits in advance. One way to do this uses a technique of logging function and system library calls as well as kernel calls [46].

BackTracker [30] uses previously recorded system calls and some assumptions about

system call dependencies to generate graphical traces of system events that have affected or have been affected by the file or process given as input. This addresses a problem of correlating data that many other logging and auditing tools have been unable to. However, an analyst using BackTracker may not know what input to provide, since suspicious files and process IDs are not easy to discover when the analysis takes place long after the intrusion. For example, identifying a file would show a graphical trace of the processes and network sockets that cause it to be there. But BackTracker does not help *identify* the starting point; it was not a stated goal of BackTracker, or its successors [53, 31]. Nor does BackTracker help to identify what happens *within* the process, because BackTracker is primarily aimed at a process-level granularity. Therefore, using BackTracker to answer questions about data that stand up as forensic evidence is not appropriate, even though the tool may appear to be perfectly accurate.

Data from intrusion detection systems has been proposed for use as legal forensic evidence [54, 58], and the admissibility and validity of the IDS data as evidence (rather than simply an early-warning system) has been studied [55]. More work is needed in this area, however. Indeed, it is not clear what admissible claims can truly be made from specific IDS-generated data.

Data describing actions on networks would seem to be simpler than data describing actions on hosts. But even here, the forensic situation is complicated. One can capture network traces, which are useful in understanding remote manipulation of systems. Sometimes even encrypted data can yield useful information about a system [32]. But sometimes tools that gather this information are simply unreliable [19]. Tools that gather network traces can drop packets—and those dropped packets could contain critical evidence. Attackers can forge packets designed to make the information that *is* collected confusing.

Data gathered from a host can be forged, too. For example, administrators can make their own syslog entries by altering the appropriate file, which is stored on a non-secure medium [52] or transmitted over a network using a non-secure protocol. How can we know that data in those logs has not been forged or tampered with?

This points to a key deficiency: the need to measure the accuracy of data before using it as scientifically valid evidence. This is a tricky problem; indeed, in some environments, it might be intractable [63]. However, this problem needs to be addressed. The scientific method must form the basis of measuring the accuracy of tools and techniques.

Earlier sections discussed how people who use systems or data from systems often make incorrect or invalid assumptions about the way the systems actually work. Similarly, many research efforts in computer science are conducted in ways that other scientific disciplines would view as not being scientifically valid [43]. That is, they fail to use techniques that have been validated using this process:

1. Define the question
2. Form a hypothesis
3. Perform an experiment and collect data
4. Analyze the data
5. Interpret the data and draw conclusions that serve as a starting point for new hypotheses
6. Publish the results (return to #3 and iterate)

Consider the work in applying biological techniques to intrusion detection [27]. Though it was ultimately shown to be valid (and has been extended by many others) some of the

data that was initially used to evaluate its effectiveness was shown to be flawed [59]. Before the eventual re-analysis of the technique, the impact such a discovery would have had on any ongoing legal case based on evidence collected by such an IDS is unclear. How much can we depend on individual software tools that have not been reviewed by disinterested parties, and whose reviews are not reproducible? Consider the problem of measuring effectiveness in intrusion detection in genera. A number of papers use the DARPA/Lincoln Labs network intrusion detection datasets as a method of evaluating their own techniques, despite the significant flaws shown in the datasets [37]. One of the reasons why researchers continue to use the datasets is because creating *new* datasets is hard, and getting them to be widely adopted (so that other, future methods can be compared directly with current ones) is perhaps even harder. Further, even those researchers who do create their own datasets often run their experiments in variable conditions, such that even if the dataset were properly captured, the experiment could not be reproduced exactly. This violates one of the fundamental tenets of scientific experiments mentioned above: reproducibility for third-party verifiability.

Once the accuracy of techniques have been established, we must develop and use models of forensic process to understand the circumstances under which it is appropriate to make claims about the data derived from the techniques.

5: Forensic Models

Forensic practitioners and computer scientists both agree that "forensic models" are important for guiding the development of forensic tools and techniques. Models generalize an ad hoc process to provide a framework that enables people to understand what that process does, and does not, do. The way in which each party use the term "model" has important differences. Practitioners define the term "model" as an abstraction of a process of examining evidence, regardless of whether the evidence is DNA or computer files [50]. The same process used for examining DNA evidence is mapped to computer files using the models. Computer scientists use "model" to mean a simplified description to assist calculations and descriptions. In computer science, a model represents an abstraction of something that contains sufficient detail to be useful as a predictive formula. Putting it bluntly (and somewhat unfairly), forensic practitioners think of models as recipes, while computer scientists think of models as simplifications of reality.

Each definition refers a predictive formula or mapping. The difference between the two is simply the level of abstraction. The computer science definition focuses most closely on linking a low-level series of events within an operating system to a higher-level event. The "calculation" refers to capturing nondeterministic events in order to predict and understand the deterministic series of events that follow. In discussing the mapping of recorded data to analyzed data, computer scientists sometimes refer to the data that is collected on a system already, and at other times refer to data the collection of which requires augmenting a system. The definition used by forensic practitioners looks at a higher level, which includes multiple steps: gathering, protecting, preserving, and analyzing data. That definition focuses on data collected from an existing system, rather than data that might be collected were the system modified. The practitioners' model emphasizes the broader forensic process because the practitioners consider the legal aspects of using the data. That is, the emphasis is not on a mathematically complete mapping and total understanding of the past events,

but on practical elements of data collection and issues of admissibility, such as preserving the chain of custody of the evidence. Ideally, a definition of "model" should include *both* a low-level view of a system and a high-level view that takes physical, procedural, and legal elements into account, too.

The tools discussed in the previous section can be used to collect data specified by forensic models. The models should allow an analyst to determine *when* using existing tools is appropriate and when to develop new tools that better fit the needs of forensic practitioners. In some sense, the hypothesis describing events in enough detail to deduce cause and effect *is* the default model of anyone using those tools. We seek a more complete hypothesis, along with a measure of accuracy.

Numerous approaches by forensic practitioners and even computer scientists focus on the elements of computer forensics that relate directly to the law. Andrew's model [5]—a "process model"—consists of two principles (consistent results and static storage) and five laws (association, context, access, intent, and validation). The principles and laws define a mapping between data and the methods by which that data is collected and analyzed, and the admissibility of that data in court. The process model also defines possible outcomes, such as "can be shown to have occurred," "can be shown to be have likely occurred," "can be shown to be unlikely to have occurred," and "can be shown to have not occurred." As an example, the law of access is: "It must be demonstrated that the individual had access to the device at the time the data was created." The law specifies ways in which that fact can be shown, but ultimately, neither the law or the model in general specify how such a requirement can be implemented, and then how the implementation can be tested and measured.

Carrier created a model that maps the physical investigative process to the digital world [14]. The steps in a physical investigation include preservation of evidence, survey for evidence, documentation of evidence, search for evidence, crime scene reconstruction, presentation of a theory. During the search for physical evidence, Carrier first discusses how the *physical* search process will also turn up *digital* evidence in the form of disks and other digital media. He then discusses how the exact same steps in the physical process are necessary in the digital world as well, and feed back to create a more thorough crime scene reconstruction. Finally, Carrier discusses at a high level how some of the elements of a digital investigation map directly back to aid a physical investigation, such as *digital* photographs that identify *real* people. Carrier's model is useful and important. It teaches computer scientists how their tools and the resulting data are likely to be used by forensic analysts in an investigation. But it has a specific goal of mapping "physical to digital," which does not address the lower-level issues of the validity of computer forensic evidence or when it is appropriate to make claims about the digital evidence itself. Carrier's model is one example of high-level models of the computer forensic process, and is representative of the goals and style of other such models. However, this type of model must be merged with a lower level computer science-based model to be complete.

Computer scientists have also tried several approaches to construct forensic models. Gross [24] studied usable data and analysis techniques from unaugmented systems. He formalized existing *auditing* techniques already used by forensic analysts. One method involved classifying system events into categories representing transitions between system states. Then, he developed methods of analyzing the differences between system states to attempt to reconstruct the actions that occurred between them, using assumptions about the transitions that must have come before. In Gross's model, broad classifications of events

included creation, deletion, storage, retrieval, query, and change. Each of those categories is represented in different scenarios in a computer system by a different mechanism. For example, in a *process*, "storage" and "retrieval" refer to swapping a process out to or in from disk, whereas when referring to an I/O device, "storage" and "retrieval" refer to sending or receiving information to or from the I/O device. By applying the classification for each scenario (processes, I/O, memory management, files, and the kernel) to an operating system, one can examine the data sources (e.g., system log messages from applications, and network connection information from TCP Wrappers) and apply the classifications to piece together the grains of the previous events.

Gross did not discuss a methodology for separating the relevant information from the rest of the system information. However, the process of building a model useful for legal purposes should address a methodology of understanding the information actually *necessary* to analyze specific, discrete events such as attacks. Most of Gross's research that focused on events focused on using *available* data that was already being collected on systems to improve analysis, and make it more efficient, as opposed to augmenting a system to collect *necessary* data that is not yet being collected. One area that did focus on augmenting systems was a model that included an entropy analysis of the relationship between a filesystem and a disk that could be used to recover previously-erased files. Both models that Gross developed *could* be highly useful to legal applications. However, on their own, they do not address goals of the models used by forensic practitioners, or the combined model that we wish to produce because they do not tell us how and when it is appropriate and valid to use data produced or recovered by the techniques in court.

Previous research in modeling systems has examined the limits of auditing in general [6] and auditing for policy enforcement [51]. However, neither of these efforts studied presenting *useful* information to a human analyst or requirements for legally admissible information. Other work [34] evaluated the effect of using different audit methods for different areas of focus (attacks, intrusions, misuse, and forensics) with different temporal divisions (real-time, near real-time, periodic, or archival), but again, the results focused primarily on performance rather than forensic value .

One framework that has focused on usability by a forensic analyst presented a mechanism that ties objects and events back to their origin [12, 11]. It binds labels to processes and system objects. This approach could be viewed as being similar to the approached used in *BackTracker* [30], but broader and more flexible. A simple example is a type of network traceback to look at a chain of `ssh` connections. The goal would be to determine the system that the chain of connections originated from, and if possible the originating process and user. The use of the model to build the mechanism differs from previous efforts to build computer forensic systems, because the process starts by asking what is necessary (i.e., data that helps to find the origin of a crime) and then helps to guide and augment the mechanisms to provide that data. The model could be a useful component of a broader model incorporating legal issues, but a lack of experimental data makes difficult gauging the actual usefulness of the model in practice.

Another framework from computer science is a scientific, hypothesis-based approach using finite state machine models to analyze events and objects in computer systems [15]. This approach links forensic data to the steps that must be taken to investigate the history of that data by answering specific questions through the observation of a controlled experiment. In this manner, the model follows the application of the scientific method. The model established multiple classes of analysis techniques that could be used to test

hypotheses about forensic questions, and a proof demonstrated the classes to be complete.

For example, "does file X exist" seems like a simple forensic question, but understanding the validity of the answer raises multiple questions. How would we observe if the file existed? What are the capabilities of the system to store such data? In practical terms, can we test that hypothesis? The application of Carrier's model to court cases has not yet been demonstrated, nor has experimental data been developed to validate the implementation of the model in practice. Nevertheless, Carrier's approach demonstrates how the scientific method can be applied to a forensic model, which we believe is ultimately a crucial element of a complete model.

Previous work has discussed many of the problems and constraints in existing computer forensics tools and developed a set of principles to address those problems [45]. Those principles have been used to develop a solution [46]. But even there, assumptions underlie claims about the completeness and effectiveness of the solution. A systematic approach that guides forensic logging and auditing is necessary [47, 49]. A structured approach is needed for two reasons. First, understanding the landscape of *possible* objects and events must precede deciding whether a solution is complete. Second, understanding the problem allows the development of the relevant *necessary* and *sufficient* objects and actions. Indeed, the model is designed to investigate intruders attempting to achieve their goals. The model employs attack graphs based on intruder goals and violations of security policy to define the information that needs to be logged on a computer system in order to forensically analyze those events.

As an example of this model, consider an intruder trying to gain remote access to a UNIX shell without properly authenticating. The 1988 Internet Worm [18], a classic multi-stage, multi-exploit attack, exploited a buffer overflow vulnerability in `fingerd`, and several problems with other UNIX programs to break into systems. The attack caused denials of service by propagating to as many machines as possible, causing the systems to be swamped and unusable. Using this model, we can show that additional data would have simplified analysis of the worm, as follows.

The ultimate goal of the worm was to spread. The worm took the following steps, which are characteristic of a broad class of worms:

1. Run multiple exploits against a system.
2. Once on the system, invoke a shell running as the user who owned the process that was exploited, or as a user whose account was compromised (either by guessing a password or through trust relationships).
3. Spawn a copy of itself approximately every 3 minutes to refresh its appearance of use.
4. Meanwhile, try to spread to other machines.

The Internet worm could be modeled at a higher or lower level of abstraction. We choose a level appropriate for our analysis in which we model the known steps and attempt to place bounds on the unknown ones—the exploits—that are impossible to predict completely and too numerous to enumerate. In this case, the model considers each stage of the worm separately after having first been triggered by the initial remote access connection. The construction of the model then drives the specific forensic data needed for analysis. Since that forensic data is rarely recorded automatically by the system, gathering it may require modification to the system (via a kernel module, etc...) to capture the data.

Using a simple set of assumptions [44], we compared the effectiveness of that model and a previous approach, and both show promise. Ultimately, however, we wish to find a way of

rigorously and scientifically evaluating both approaches as well as the approaches of other existing and future forensic tools under the proper environment [43], which we hope will include legal applications.

None of the practitioners' models address forensic logging and auditing at a low enough level of abstraction that systems can be practically implemented using those models alone. Likewise, none of the computer science models—even those that focus on scientific method and addressing specific legal questions—can be applied to address the problems that make computer forensic data inaccurate or often invalid, due to open questions about the accuracy of measurements and/or validity about the meaning of the data.

One method of integrating models of different foci and levels of granularity is to first understand the needs of law enforcement, then apply a process model approach [14] to drive the understanding of intruder goals and a forensic logging system based on attack graphs [49]. Ultimately, one could use a hypothesis-based model [15] that uses the scientific method [43] to enhance the theoretical model and verify completeness. But the details of this method are complicated, and the solution is not so simple as this example suggests. However, it is an example of the kind of process and tools that must be used to develop a more complete solution.

Integrating the models, understanding the goals needs of law enforcement, and the capabilities of forensic systems will also lead to defining metrics to understand the value of forensic data. For example, how do we measure how well forensic methodologies and tools, capture data? Metrics should guide the nature and amount of data necessary to validate the metric, or refine it. One aspect of such a metric could be how well it maps the forensic data back to a particular attack or set of attacks. Ideally, each forensic "trace" should correspond to a single attack or vulnerability. However, if this is not possible, one could focus on metrics that simply minimize the *set* of possible attacks. This also helps refine the tools and techniques used to uncover the data. For example, suppose that three types of data from separate data sources are gathered, and from that data, it can be concluded that one of ten possible attacks was used. If it is possible to add one more type of data and reduce the number of possible attacks to two, but adding two other types of data reduce the number of attacks to three, the first is preferable in most cases. This suggests an obvious metric.

But the metric may be misleading. Perhaps the second case allows one more type of data to be added and helps to determine which of the three attacks were used, but the first case requires three more types of data to determine which of the two attacks were used. This suggests the metric should favor the second case over the first. Therefore, measuring the accuracy of forensic tools and technologies requires that the tools be related to the forensic evidence which they are to uncover. To understand the limits of these tools, one must also understand the nature of that which they measure, attacks and vulnerabilities.

The synthesis of the computer scientists' forensic models with the forensic practitioners' forensic models will ultimately result in a more complete and useful method for mapping the forensic data to the systems that are built to identify, collect, preserve, examine, analyze, and present it, and ultimately produce a legal decision. Undoubtedly, some of the tools suggested by the model will exist. For example, TCP Wrappers is a robust and reliable tool for collecting data about TCP network connections. This is important information in many models, and so TCP Wrappers is a reasonable tool to consider using when models require this data. At other times, the models will require data that existing tools do not gather, and then either new tools must be created or existing tools must be enhanced,

to gather the required data. For example, a model may specify some requirements for securing the integrity and confidentiality of audit logs. TCP Wrappers does not do that on its own. The computer science-based forensic models demonstrate that it is possible to determine accuracy of logging and analysis methods by suggesting metrics for evaluating those methods.

But the models still do not indicate when the data is not valid. Biologists know how DNA evidence can be contaminated and what the relationships are between DNA taken from one individual and that individual's identical twin, other siblings, and parents. But computer science models ignore key questions that involve human procedure and how data is really created. Is a file on the system because Alice downloaded it or because Bob planted it? How can we tell the difference between Alice's actions and Bob's? The computer science models also ignore components of the most critical components of the rules of evidence that govern admissibility, such as ensuring that there exists a mechanism that records it with complete accuracy and that once stored, the data is inviolate.

6: Case Studies

6.1: Case Study #1: Gates v. Bando

We now present a high-level case study that demonstrates the problems that arise from considering only technical or only procedural uses of forensic data. In 1992, the Gates Rubber Company ("Gates") accused Robert Newman, a former employee, of violating Gates's intellectual property rights by stealing a computer program called "Life in Hours" when he left the employment of Gates and went to work at Bando Chemical Industries ("Bando") [23, 54].

Prior to going to trial, Gates obtained the hard drive from the system that they claimed the stolen software was used on. After analyzing the drive, Gates alleged that Newman had destroyed evidence by deleting the Life in Hours software from his system at Bando *after* being accused, along with documents created with Life in Hours (which would have implied that the software had been used). In particular, Gates pointed to a directory containing 45 file entries on the disk provided to Gates, but only 44 file entries on the disk provided to the court. Gates argued that the missing file (eventually recovered, because it was "erased" but not "wiped") demonstrated destruction of evidence as well as an attempt to mislead the court. Bando claimed to not know how the discrepancy could have occurred.

Eventually, Newman admitted that he had used a program to "clean up" his word processing files, but did not intentionally delete anything. The expert hired by Bando (Dr. Robert Wedig) pointed out that the "cleaning" program could have unintentionally deleted the files. Wedig also questioned how the discrepancy could have helped Bando, and therefore what reason Bando would have had for intentionally misleading Gates or the court. Finally, Wedig noted procedural errors made by the prosecution's expert (Robert Voorhees):

1. The program that used to recover the files, Norton Unerase, was copied to and run on the target disk. In doing so, 7-8% of the disk was overwritten.
2. All analysis should have been performed on a read-only backup *image* so that no possible tampering or alteration of the evidence could have occurred.
3. The creation dates of the files that overwrote the deleted files were not identified, and it could not be determined whether the files were deleted before or after the suit.

In 1992, procedural standards for preserving evidence had not been established and technical issues relating to deletion, creation, and overwriting files on disk were not well understood by courts. These lack of standards led Voorhees to misinterpret the data, and caused Gates to spend millions of dollars and several years to pursue flawed litigation.

The problems that occurred in Gates v. Bando are still quite prevalent, as demonstrated in the Amero case (from Section 1) and many others. Though few cases involving digital evidence go to court today without efforts taken to preserve the evidence, a synthesis of procedure and technological understanding continues to lead to misinterpretation of data. Unfortunately courts and many 'experts' still have not made the transition to thinking of computers as evidence. The goal of obtaining *provable* facts, as opposed to *probable* facts has still not been realized for the purposes of litigation [54].

This case presents a clear scenario where understanding both technological and procedural (computer science and legal) requirements must be understood, merged, modeled, implemented, and followed for forensic data to be useful and used. In this case, the use of a model such as Laocoön could have helped to indicate the technological threats and a model of procedure could have helped to indicate the procedural threats. The integration of the two models could have helped either side (including the defense) identify the possible origin of the files, by helping to identify the assumptions involved: in this case, the incorrect assumption that the disk that was analyzed by the prosecution had been untainted prior to its admission into court as evidence in the Gates v. Bando case.

6.2: Case Study #2: Electronic Voting

We now present a case study involving electronic voting that demonstrates the problems that arise from considering only technical or only procedural uses of forensic data.

In Florida, the election for the Congressional District 13 (CD13) showed an anomaly: the number of undervotes[5] was an order of magnitude higher than expected for such a high-profile, contentious race. Further, the same anomaly occurred in only one other race out of 25. As there were no VVPATs associated with the machines, the dispute was whether the voting machines correctly recorded the votes that the voters intended to cast. Some believed that the voters did indeed cast their votes for the race and so the source of the undervotes was malfunctioning voting machines. Others thought that the contentiousness of the race had caused voters to skip that particular race. A widely accepted explanation is that poor ballot design caused voters to miss the CD13 race on the ballot. Still, there is little physical evidence to support any hypothesis.

The State of Florida audited the election. Part of that audit involved dynamic testing of the voting machines to see if any problems not revealed before the election arose in this post- election testing. A second part involved a source code audit of the voting system to determine if any software vulnerabilities or flaws could have caused or contributed to the undervotes. The audit concluded that the examined software did not cause or contribute to that problem [65].

During the analysis, the investigators commented that if the machines had a VVPAT, that "audit trail" would have been of little help in the analysis. If the VVPAT showed the undervote, it would be echoing the current results, and again the question would arise whether the undervote was due to the voter not casting a vote for the particular race, or was due to a programming error: exactly the situation without a VVPAT. However, if the

[5] An undervote is when a ballot is cast with no valid selection for a particular race.

VVPAT showed the voter having cast a vote for the race and the electronic count showed an undervote, then the investigators would know that an inconsistency occurred either before the vote was printed (in the case where a voter did not vote, and the VVPAT said she had) or after he vote was printed (in the case where a voter voted, and the electronic record of the ballot said she had not). Thus, the VVPAT could have confirmed that a problem existed without providing any guidance on where the problem might be. It could not have demonstrated that no problem existed.

So what auditing mechanisms would be useful to provide the information necessary to audit the system in a way that would uncover problems such as this? Specifically, what are the precise points in the system at which audit mechanisms should be placed and what data should those audit mechanisms record? Properly using appropriate data gathering mechanisms would enhance both security and integrity, because they could be used to track events on the system to detect malicious activity (security) as well as tampering, alterations, and errors involving the recording and counting of votes (integrity). We call the log generated by these mechanisms a forensic audit trail, or FAT, to distinguish it from the voter-verified audit trail (VVPAT).

But collecting a FAT also leads to a problem. Consider George, who wants to sell his vote. John says that he will pay George if George votes for a certain Presidential candidate named Thomas. John tells George to vote for Thomas by first voting for James, then changing his vote to Andrew, then back to James, then to Thomas. This sequence of votes (James/Andrew/James/Thomas) is highly unlikely to be accidental. If John sees this sequence in a FAT, he can guess that George probably cast that vote. This type of communication uses a covert channel, or a path through which people can communicate but that was not designed for such communication. This type of messaging is anathema to the U.S. election system in general because it enables the selling of votes. Any FAT must guard against this problem.

We believe that a model such as Laocoön [49] is particularly well suited for critical systems, such as electronic voting machines for several reasons. First, electronic voting machines have limited modes of operation compared to general purpose computing systems. Second, electronic voting machines have well-defined security policies compared to general purpose computing systems. For example, many possible violations of security policy on an electronic voting machine (e.g., changing or discarding cast votes) are easy to define precisely. The benefit of the model is that deriving or knowing the methods for violating the policieswhich is known to be difficultare not necessary to pre- define. The result is that only the information that is necessary to analyze to understand the violations or suspected violations is recorded. Without the model, system designers would either have to guess about what is important (creating the risk of missing something) or record everything (thus overwhelming both the auditing system and the human analyst).

However, most importantly (and most difficult) in this case, Laocoön provides a technological framework for balancing balance secrecy and anonymity needs with auditing. Protecting and secrecy and anonymity is the law in many states in the United States. By using Laocoön to first systematically characterize both attacker and defender goals, one can then more easily characterize the threats and targets. By understanding the context of attacks and failures better, an analyst can weigh forensic metrics against privacy metrics to evaluate what information can be revealed to the auditors and therefore what, of the requirements given by Laocoön, can and cannot be safely recorded. In this manner, if a particular state does not want auditors to be able to see how voters cast their ballots,

the model applies one set of constraints. Alternatively, if a state has no such laws, determines that that the auditors are trusted, or establishes procedural mechanisms to provide protection to preserve anonymity, a different set of privacy constraints could be applied.

However, it is also clear that Laocoön alone is insufficient, and needs to be combined with a procedural model. For example, the system should be built in such a way that the logging mechanism cannot be bypassed and the logs are tamperproof, or measurably tamper-resistant. This requires the use of a reference monitor-like component and write-once storage devices. However, even such low-level mechanisms are not 100% reliable, and so they also must be designed to fail in a safe way.

For example, one way to store the logs is to use a transaction-based system that records each transaction to a separate system with a separate (and perhaps even more rigorous) security domain. A transaction on the primary system cannot proceed without acknowledgment of receipt by the logging system. However, if the link between the two is broken, the primary system should stop functioning. This is vulnerable to a denial-of-service attack, of course, but certain jurisdictions may find this more desirable than over-votes or other actions proceeding without being logged.

And, regardless, the model must also account for procedural elements that humans must follow outside of any technical context. Therefore, the design of an audit system should begin with voting machine requirements, which we have previous discussed. The research questions that we previously posed also drive the construction of the forensic model by giving a starting point for understanding what the system looks like when it's running as it is intended, when it is running in violation of its original intention, and what elements of the system could cause a transition between those two state spaces.

7: Conclusions

The need for a common language for computer forensics is clear. Computer forensics lags behind other forensic disciplines in part due to insufficient dialogue between researchers and practitioners, and the result is that *science*—a fundamental component of forensics—is largely absent from computer forensics. An ability to communicate about the challenges of each side will ultimately help bring scientific method [43, 44] to computer forensics in the way that it exists in other forensic disciplines, such as DNA analysis where the statistics and science regarding the accuracy of the tests is well-understood.

The problems that can be answered through the collaboration of forensic practitioners and computer scientists by understanding each other's goals, building complete models of systems and procedures, and then implementing those systems and following the procedures, include the following:

1. How accurate is the method used to produce the data?
2. How accurate is the method used to analyze the data?
3. What claims can be made about the data?
4. What assumptions must be made to make those claims?
5. What can we do to reduce the amount of assumptions that must be made to use the data?

We cannot achieve perfection in a world that requires interaction with and interpretation by humans. None of the computer systems will stand on their own without a thorough

analysis of each system to understand and define the limits of the technology and which human procedures support it. But it is essential that we take the realistic steps to contribute to legal systems by way of accurate and valid forensic tools, contributions to democracy by way of designing and implementing unassailable voting systems, and contributions to other disciplines and industries that assume that the work of computer scientists has been scientifically supported to a much greater extent than it actually has.

8: Acknowledgements

The authors wish to thank Fred Chris Smith for his help in understanding the legal issues that we present in this paper, and also Becky Bace, Todd Heberlein, Michael Losavio, and Chris Wee for inspiring us to think outside the box about the many, varied uses of forensics in the "real" world.

References

[1] *New Oxford American Dictionary*. Oxford University Press, Second edition, May 2005.

[2] R. Abdulrahim. Results of Goshen school vote on $70M bond are lost forever. *Times Herald-Record*, December 6 2007.

[3] E. Allman. Personal conversations, January 2005.

[4] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, April 1980.

[5] M. W. Andrew. Defining a Process Model for Forensic Analysis of Digital Devices and Storage Media. In *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 16–30, Seattle, WA, April 2007.

[6] M. Bishop. A Model of Security Monitoring. In *Proceedings of the Fifth Annual Computer Security Applications Conference (ACSAC)*, pages 46–52, Tucson, AZ, December 1989.

[7] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley Professional, Boston, MA, 2003.

[8] M. Bishop and D. Wagner. Risks of E-Voting. *Communications of the ACM*, 50(11):120, November 2008.

[9] M. Bishop *et al. UC Red Team Report of California Secretary of State Top-to-Bottom Voting Systems Review*, 2007.

[10] D. Bonyun. The Role of a Well-Defined Auditing Process in the Enforcement of Privacy and Data Security. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, 1980.

[11] F. Buchholz. *Pervasive Binding of Labels to System Processes*. PhD thesis, Purdue University, 2005.

[12] F. P. Buchholz and C. Shields. Providing process origin information to aid in computer forensic investigations. *Journal of Computer Security*, 12(5):753–776, September 2004.

[13] B. Carrier. *File System Forensic Analysis*. Addison Wesley Professional, 2005.

[14] B. Carrier and E. H. Spafford. Getting Physical with the Digital Investigation Process. *International Journal of Digital Evidence*, 2(2), Fall 2003.

[15] B. D. Carrier. *A Hypothesis-Based Approach to Digital Forensic Investigations*. PhD thesis, Purdue University, 2006.

[16] A. L. Cowan. Teacher Faces Jail Over Pornography on Class Computer. *New York Times*, February 14 2007.

[17] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In *Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[18] M. W. Eichin and J. A. Rochlis. With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, Oakland, CA, 1989.

[19] B. E. Endicott-Popovsky, J. D. Fluckiger, and D. A. Frincke. Establishing Tap Reliability in Expert Witness Testimony: Using Scenarios to Identify Calibration Needs. In *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 131–144, Seattle, WA, April 2007.

[20] D. Farmer and W. Venema. *Forensic Discovery*. Addison Wesley Professional, 2004.

[21] R. P. Feynman. *The Feynman Lectures on Physics*, volume 1, chapter 1: Atoms in Motion. Addison-Wesley, 1964.

[22] R. Gardner, S. Garera, and A. D. Rubin. On the difficulty of validating voting machine software with software. In *Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*, pages 39–54, 2007.

[23] Gates Rubber Comany v. Bando Chemical Industries, Ltd., et al. 167 F.R.D. 90, U.S. Dist.,. Lexis Nexis 12423, Decided May 1 1996.

[24] A. H. Gross. *Analyzing Computer Intrusions*. PhD thesis, University of California, San Diego, Department of Electrical and Computer Engineering, 1997.

[25] P. Gutmann. Data Remanence in Semiconductor Devices. In *Proceedings of the 10th USENIX Security Symposium*, 2001.

[26] D. Hayes (quoting Scott C. Williams, supervisory special agent for the FBI's computer analysis and response team in Kansas City). KC to join high-tech fight against high-tech crimes: FBI to open $2 million center here. *Kansas City Star*, page A1, April 26 2002.

[27] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion Detection using Sequences of System Calls. *Journal of Computer Security*, 6:151–180, 1999.

[28] E. Kenneally. Computer Forensics Beyond the Buzzword. *;login:*, 27(4):8–11, August 2002.

[29] G. H. Kim and E. H. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In *Proceedings of the 1994 ACM Conference on Communications and Computer Security (CCS)*, November 1994.

[30] S. T. King and P. M. Chen. Backtracking Intrusions. *ACM Transactions on Computer Systems*, 23(1):51–76, February 2005.

[31] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen. Enriching Intrusion Alerts Through Multi-Host Causality. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS)*, 2005.

[32] T. Kohno, A. Broido, and kc claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2(2):93–108, April–June 2005.

[33] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an Electronic Voting System. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 27–40, 2004.

[34] B. A. Kuperman. *A Categorization of Computer Security Monitoring Systems and the Impact on the Design of Audit Sources*. PhD thesis, Purdue University, 2004.

[35] U. Lindqvist and P. A. Porras. eXpert-BSM: A Host-Based Intrusion Detection System for Sun Solaris. In *Proceedings of the 17th Annual Computer Secrity Applications Conference (ACSAC)*, pages 240–251. IEEE Computer Society, December 10–14 2001.

[36] Martin Littlefield (*Assistant U.S. Attorney, WDNY*). Emerging Legal and Forensic Issues for Computer Scientists Teaching Digital Forensics and Information Assurance: The Import of *United States v. Garnier*. Digital Forensics Working Group 2007 Workshop, June 2007.

[37] J. McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by the Lincoln Laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, November 2000.

[38] S. Narayanasamy, G. Pokam, and B. Calder. BugNet: Continuously Recording Program Execution for Deterministic Replay Debugging. In *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA)*, June 2005.

[39] National Institute of Standards and Technology (NIST). Computer forensic tool testing program. `http://www.cftt.nist.gov/`.

[40] National Institute of Standards and Technology (NIST). Deleted File Recovery Specifications Draft Report. `http://www.cftt.nist.gov/DFR-Specification-SC.pdf`, January 19 2005.

[41] W. Osser and A. Noordergraaf. *Auditing in the Solaris Operating Environment*. Sun Microsystems, Inc., February 2001.

[42] S. Peisert. Forensics for System Administrators. *;login:*, 30(4):34–42, August 2005.

[43] S. Peisert and M. Bishop. How to Design Computer Security Experiments. In *Proceedings of the Fifth World Conference on Information Security Education (WISE)*, pages 141–148, West Point, NY, June 2007.

[44] S. Peisert and M. Bishop. I'm a Scientist, Not a Philosopher! *IEEE Security and Privacy Magazine*, 5(4):48–51, July–August 2007.

[45] S. Peisert, M. Bishop, S. Karin, and K. Marzullo. Principles-Driven Forensic Analysis. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 85–93, Lake Arrowhead, CA, October 2005.

[46] S. Peisert, M. Bishop, S. Karin, and K. Marzullo. Analysis of Computer Intrusions Using Sequences of Function Calls. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(2):137–150, April–June 2007.

[47] S. Peisert, M. Bishop, S. Karin, and K. Marzullo. Toward Models for Forensic Analysis. In *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 3–15, Seattle, WA, April 2007.

[48] S. Peisert, M. Bishop, and K. Marzullo. Computer Forensics *In Forensis*. *ACM Operating Systems Review (OSR) Special Issue on Computer Forensics*, 42(2), April 2008.

[49] S. P. Peisert. *A Model of Forensic Analysis Using Goal-Oriented Logging*. PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, March 2007.

[50] M. M. Pollitt. An Ad Hoc Review of Digital Forensic Models. In *Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 43–52, Seattle, WA, April 2007.

[51] F. B. Schneider. Enforceable Security Policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1):30–50, February 2000.

[52] B. Schneier and J. Kelsey. Secure Audit Logs to Support Computer Forensics. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):159–176, May 1999.

[53] S. Sitaraman and S. Venkatesan. Forensic Analysis of File System Intrusions using Improved Backtracking. In *Proceedings of the Third IEEE International Workshop on Information Assurance*, pages 154–163, 2005.

[54] F. C. Smith and R. G. Bace. *A Guide to Forensic Testimony: The Art and Practice of Presenting Testimony As An Expert Technical Witness*. Addison Wesley Professional, 2003.

[55] P. Sommer. Intrusion Detection Systems as Evidence. In *Proceedings of the First International Workshop on Recent Advances in Intrusion Detection (RAID)*, 1998.

[56] E. H. Spafford and S. A. Weeber. Software forensics: Can we track code to its authors? Technical Report CSD-TR 92-010, Department of Computer Science, Purdue University, 1992.

[57] T. Stallard and K. Levitt. Automated Analysis for Digital Forensic Science: Semantic Integrity Checking. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, December 8-12 2003.

[58] P. Stephenson. The Application of Intrusion Detection Systems in a Forensic Environment (extended abstract). In *The Third International Workshop on Recent Advances in Intrusion Detection (RAID)*, 2000.

[59] K. M. Tan and R. A. Maxion. "Why 6?" — Defining the Operational Limits of stide, an Anomaly-Based Intrusion Detector. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 188–201, Oakland, CA, 2002.

[60] K. Thompson. Reflections on Trusting Trust. *Communications of the ACM*, 27(8):761–763, August 1984.

[61] W. Venema. TCP WRAPPER: Network monitoring, access control, and booby traps. In *Proceedings of the 3rd USENIX Security Symposium*, September 1992.

[62] E. J. Wagner. *The Science of Sherlock Holmes*. Wiley, 2006.

[63] J. Wildermuth. Secretary of state casts doubt on future of electronic voting. *San Francisco Chronicle*, pages C–7, December 2 2007.

[64] A. Yasinsac and M. Bishop. Of Paper Trails and Voter Receipts. In *Proceedings of the 2008 Hawaii International Conference on System Sciences*, January 2008.

[65] A. Yasinsac, D. Wagner, M. Bishop, T. Baker, B. de Medeiros, G. Tyson, M. Shamos, and M. Burmester. *Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware: Final Report For the Florida Department of State*. Security and Assurance in Information Technology Laboratory, Florida State University, Tallahassee, Florida, February 23 2007.

[66] K. J. Ziese. Computer based forensics – a case study – U.S. support to the U.N. In *Proceedings of CMAD IV: Computer Misuse and Anomaly Detection*, 1996.