56

Digital Forensics

56.1 Introduction .........................................................56-1
56.2 Definitions ..............................................................56-2
56.3 Inquiry Models and Tool Uses......................................56-3
  Pirolli and Card Cognitive Model • Model Relationships • Mathematical Models
56.4 System Analysis .......................................................56-6
  Storage Forensics • Memory Forensics • Network Forensics
56.5 Artifact Analysis ......................................................56-16
  Crypto Hashing • Approximate Matching • Multimedia • Event/Timeline Reconstruction • Application Forensics
56.6 Forensic Tool Design and Implementation .....................56-21
  Scalability and Automation • Standards and Inter-Operability • Tool Testing and Calibration
56.7 Summary ...................................................................56-23
56.8 Open Issues and Challenges ........................................56-23
  Scalability and Automation • Sensor and Device Diversity • Virtualization and Cloud Computing • Pervasive Encryption • Legal Issues and Proactive Forensics

Glossary ........................................................................56-24
References .......................................................................56-25

Vassil Roussev
University of New Orleans

56.1 Introduction

Forensic science (or forensics) is dedicated to the systematic application of scientific methods to gather and analyze evidence in order to establish facts and conclusions that can be presented in a legal proceeding. Digital forensics (a.k.a. computer or cyber forensics) is a subfield within forensics, which deals specifically with digital artifacts, such as files, and computer systems and networks used to create, transform, transmit, and store them. This is a very broad definition, and there are several closely related fields and sub-fields that share methods and tools with forensics. Indeed, the forensic methods used were originally developed for a different purpose and continue to be shared. Thus, it is primarily the purpose, and not their functionality, that classify particular tools and techniques as forensic.

Malware analysis \cite{Mali08} (or malware forensics) specializes in understanding the workings and origins of malicious software, such as viruses, worms, trojans, backdoors, rootkits, and bots. Such analysis is often necessary after a security breach to understand the exploited vulnerability vector and the scope of the breach. The analysis depends on well-prepared and executed incident response phase, which identifies and collects the source data for the analysis, while restoring the affected IT systems to a known good state.

Reverse engineering, the process of inferring the structure, or run time behavior of a digital artifact, is almost always a critical step in any kind of forensic analysis. Apart from the obvious example...
of malware, most closed-source software produces artifacts that are rarely publicly documented by the vendor (unless they want to establish them as a standard). For example, the on-disk structure of Microsoft’s NTFS file system has never been officially published; the format of the files used by Microsoft Office was not officially released until 2008.*

Data recovery, the process of salvaging useable digital artifacts from partial or corrupted media, file systems, or individual files is another technique frequently employed in forensics during the initial, data acquisition, stage. More broadly, the definition of data recovery often includes the retrieval of encrypted, or otherwise hidden, data.

The intelligence community uses document and media exploitation (or DOMEX) [Garf07b] to extract actionable intelligence from relevant data sources. In practice, the DOMEX relies heavily on available forensic techniques, and higher-level analysis, to make sense out of large amounts of raw data.

The goal of this chapter is to present an overview of the state of the art in digital forensics from a technical perspective, with an emphasis on research problems. This allows us to treat the topic in a more holistic manner by presenting issues, tools, and methodologies that are common to a broader field in computer security and IT management. The legal aspects of computer forensics, which is clearly of concern to practitioners in law enforcement, can largely be modeled as placing various restrictions on the ways in which the data can be gathered and the way conclusion can be presented in a courtroom. These do present some additional research and development problems; however, these are relatively minor.

56.2 Definitions

The overall objective of digital forensic analysis is to reconstruct a chain of events that have resulted in the currently observable state of a computer system or digital artifact. The purpose of the analysis is to help answer six basic questions that arise in most inquiries: what happened? where did it happen? when did it happen? how did it happen? who did it? why did they do it? Any notion of completeness of the analysis, which can be of legal significance, is tied to the target and purposes of the original investigation and is very difficult to define in technical terms.

Another approach to understanding cyber forensics is to view it from a procedural standpoint—what does a typical investigation entail? This is an important question as one of the cornerstones of scientific processes is that the observed results must be independently reproducible. There are four generally accepted [Kent06] steps in the forensics process: collection, extraction, analysis, and reporting. Examiner preparation, in which the investigator is trained to perform the specific type of investigation, is sometimes referenced as an added, preliminary step.

During the collection phase, the investigation must identify potential sources of evidence (computers, mobile phones, flash/optical media, etc.) and choose, acquire, and preserve the appropriate ones based on the objective of the inquiry and legal restrictions. We will largely ignore this phase as it is an issue of procedure and training, not science. Similarly, the reporting phase during which investigators prepares a report and testifies will also be outside the scope of this discussion.

The main focus will be on the extraction and analysis phases, which constitute the most technically challenging parts. The investigator goes through the process of forming an initial hypothesis, searching for evidence that supports/disproves it, refining/redefining the hypothesis and going back to the data for proof. This process goes through multiple iterations, until the analyst has reconstructed the sequence of relevant events with the necessary level of certainty.

Throughout, examiners must carefully preserve the evidence and document the tools used and actions taken to demonstrate the reproducibility of their analysis.

For the remainder of the chapter, we look at the investigative process from four different perspectives in an effort to provide a comprehensive view of the field. First, we look at general models and best

practices of digital forensic inquiries that have slowly emerged over the years and can serve as a useful guide as to how a particular type of investigation should be performed.

Next, we consider forensic methods as they relate to main system components of a general purpose computing infrastructure—secondary storage, main memory, and networks—as well as the additional challenges posed by various integrated devices. Next, we consider the major types of artifacts that are typically of interest to an investigation—text, multimedia data (audio/video), executable code, and system data. Most of these have standardized (file) container formats, which allows their analysis to be decoupled from that of the source. Analytic methods in this domain can readily borrow from other fields like information retrieval and signal processing.

Finally, we consider the specific requirements that digital forensic tool developers face. Some of these, such as the efficient use of commodity parallel computing, are shared with other application domains, while others, like the potential need to comply with legal restrictions, are very specific to forensics.

56.3 Inquiry Models and Tool Uses

Over the past decade, both researchers and practitioners have been working to summarize, classify, and formalize the understanding of digital forensic tools and processes. In broad terms, models coming from practitioners, such as investigators in forensic labs, tend to describe the process as a collection of detailed procedures for routine operations (e.g., disk cloning) and high-level “best practices”. Work coming from researchers tends to focus on technical aspects of how to deal with specific problems—data recovery, search and query, etc. These models are difficult to compare with each other as they effectively speak to different parts of the problem and in different languages.

To present a unified picture of the digital forensic process, we adapt the sense-making process originally developed by Pirolli and Card [Piro05] to describe intelligence analysis. This is a cognitive model derived from an in-depth cognitive task analysis (CTA), which provides an overall view of an intelligence analyst’s work. Although some of the tools used may vary, forensic analysis is very similar in nature to intelligence analysis—in both cases analysts have to go through a mounting of raw data to identify (relatively few) relevant facts and put them together in a coherent story.

The benefit of using the Pirolli and Card model is at least threefold: (a) it provides a fairly accurate description of the investigative process in its own right and allows us to map the various tools to the different phases of the investigation; (b) it provides a suitable framework for explaining the relationships of the various models developed within the area of digital forensics; and (c) it can seamlessly incorporate information from other lines of the investigation.

56.3.1 Pirolli and Card Cognitive Model

The overall process is shown in Figure 56.1. The rectangular boxes represent different stages in the information processing pipeline, starting with raw data and ending with presentable results. Arrows indicate transformational processes that move information from one box to another. The x axis approximates the overall level of effort to move information from raw to the specific processing stage. The y axis shows the amount of structure (with respect to the investigative process) in the processed information for every stage. Thus, the overall trend is to move the relevant information from the lower left to the upper right corner of the diagram. In reality, the processing can both meander through multiple iterations of local loops, and to jump over phases (for routine cases handled by an experienced investigator).

External data sources include all sources of potential evidence for the specific investigation, such as disk images, memory snapshots, network captures, as well as reference databases, such as hashes of known files. The shoebox is a subset of all the data that has been identified as potentially relevant, such as all the email communication between two persons of interest. At any given time, the contents of the shoebox can be viewed as the analyst’s approximation of the information content potentially relevant to the case. The evidence file contains only the parts that directly speak to the case, such as specific email
Exchanges on the topic of interest. The schema contains a more organized version of the evidence, such as a timeline of events, or a graph of relationships, which allows higher-level reasoning over the evidence. A hypothesis is a tentative conclusion that explains the observed evidence in the schema and, by extension, could form the final conclusion. Once the analyst is satisfied that the hypothesis is supported by the evidence, the hypothesis turns into a presentation, which is the final product of the process. The presentation usually takes on the form of an investigator’s report that both speaks to the high-level conclusions relevant to the legal case, and also documents those low-level technical steps based on which the conclusion has been formed.

The overall analytical process is split into two main activities loops: (1) a foraging loop that involves actions taken to find potential sources of information, query them, and filter them for relevance; (2) a sense-making loop in which the analyst develops—in an iterative fashion—a conceptual model that is supported by the evidence. The information transformation processes in the two loops can be classified into bottom-up (organizing data to build a theory) or top-down (finding data based on a theory) ones. In practice, analysts apply these in an opportunistic fashion with many iterations.

56.3.1.1 Bottom-Up Processes

Search and filter: External data sources, hard disks, network traffic, etc. are searched for relevant data based on keywords, time and other constraints, in an effort to eliminate the vast majority of the data that are irrelevant.

Read and extract: Collections in the shoebox are analyzed to extract individual facts and relationships that can support or disprove a theory. The resulting pieces of artifacts (e.g., individual email messages) are usually annotated with their relevance to the case.
**Schematize:** At this step, individual facts and simple implications are organized into a schema that can help organize and help identify the significance and relationship among a growing number of facts and events. Timeline analysis is one of the basic tools of the trade; however, any method of organizing and visualizing the facts—graphs, charts, and diagrams—can greatly speed up the analysis. This is not an easy process to formalize, and most forensic tools do not directly support it. Therefore, the resulting schemas may exist on a piece of paper, or on a whiteboard, or only the mind of the investigator. Since the overall case could be quite complicated, individual schemas may cover specific aspects of it, such as the sequence of events discovered.

**Build case:** Out of the analysis of the schemas, the analyst eventually comes up with testable theories that can explain the evidence. A theory is a tentative conclusion and often requires more supporting evidence, as well as testing against alternative explanations.

**Tell story:** The typical result of a forensic investigation is a final report and, perhaps, an oral presentation in court. The actual presentation may only contain the part of the story that is strongly supported by the digital evidence; weaker points may be established by drawing on evidence from other sources.

### 56.3.1.2 Top-Down Processes

**Re-evaluate:** Feedback from clients may necessitate re-evaluations, such as the collection of stronger evidence, or the pursuit of alternative theories.

**Search for support:** A hypothesis may need more facts to be of interest and, ideally, would be tested against alternative explanations.

**Search for evidence:** Analysis of theories may require the re-evaluation of evidence to ascertain its significance/provenance, or may trigger the search for better evidence.

**Search for relations:** Pieces of evidence in the file can suggest new searches for facts and relations on the data.

**Search for information:** The feedback loop from any of the higher levels can ultimately cascade into a search for additional information—that may include new sources, or the re-examination of information that was filtered out during previous passes.

### 56.3.1.3 Foraging Loop

It has been observed [Patt01] that analysts tend to start with a high-recall/low-selectivity query, which encompassed a fairly large set of documents—many more than the analyst can afford to read. The original set is then successively modified and narrowed down before the documents are read and analyzed.

The foraging loop is a balancing act between three kinds of processing that an analyst can perform—explore, enrich, and exploit. Exploration effectively expands the shoebox by including larger amounts of data; enrichment shrinks it by providing more specific queries that include fewer objects for consideration; exploitation is the careful reading and analysis of an artifact to extract facts and inferences. Each of these options has varying cost and potential rewards and, according to information foraging theory [Piro09], analysts seek to optimize their cost/benefit trade-off.

### 56.3.1.4 Sense-Making Loop

Sense-making is a cognitive term and, according to Klein’s [Klei06] widely quoted definition, is the ability to make sense of an ambiguous situation. It is the process of creating situational awareness and understanding to support decision making under uncertainty—an effort to understand connections among people, places and events in order to anticipate their trajectories and act effectively.

There are three main processes that are involved in the sense-making loop: problem structuring—creation and exploration of hypotheses, evidentiary reasoning—the employment of evidence to support/disprove hypothesis, and decision making—selecting a course of action from a set of available alternatives.
56.3.2 Model Relationships

Using the previously mentioned cognitive model, we can map different conceptual approaches to modeling the forensic process. **Mathematical** models are suitable for relatively small components of investigative process. They can formalize individual steps in the analysis, such as inferring specific events based on the observed state of the system, or can be used to formalize hypotheses and test them using statistical means. Thus, mathematical models can support specific steps—either bottom-up or top-down—in the analytical process, but it is infeasible to model the entire investigation.

**Procedural** models come from practitioners and tend to be very linear, essentially, prescribing an overall march from the lower left to the upper right corner (Figure 56.1) while applying the available tools. These are not true scientific models but compilations of best practices based on experience.

**Legal** models of forensics come from the consumers of the final product of the analysis—legal experts—who have little, if any, technical expertise. They can be viewed as imposing boundary conditions on the process: they restrict what part of the available data may be investigated (e.g., no privileged communication), and formulate the required end result requirements (e.g., prove that the suspect was knowingly in possession of contraband material, and had the intent to distribute it). Despite some early efforts, such by Xu et al. [Xu09], these requirements and restrictions are too arbitrary and case-specific to be efficiently translated into technical requirements.

56.3.3 Mathematical Models

An alternative to the Pirolli and Card (P&C) cognitive model is to present the forensic process as reconstructing the history of a computation using mathematical formalisms. Carrier proposed one such model [Carr06a] which makes the assumption that the computer system being investigated can be reduced to a finite state machine (FSM) with set of states $Q$. The transition function is the mapping between states in $Q$ for each event in the event set, and the machine state changes only as a result of an event.

The point of Carrier’s model is not to reduce a system to an enormous FSM but to find basic building blocks from which to build higher-level constructs, while establishing properties using formal methods. In this model, one could formulate a formal hypothesis and test it against the captured state and the known capabilities (transitions) of the system. The model is lower-level than procedural frameworks and analytic taxonomies; therefore, it offers the promise of providing formal proofs to show the completeness of the analytic techniques employed.

Statistical approaches have the potential to be more practical by enabling formal hypothesis testing and quantifying the level of confidence in the conclusions. For example, Kwan et al. [Kwan08] and [Over10] et al. show how Bayesian networks can be used to evaluate the strength of the evidence for some simple cases and study the sensitivity of the model to inputs. The primary challenge, however, is providing reliable estimates of the various probabilities upon which the model depends.

56.4 System Analysis

In this section we outline the various approaches to acquiring and analyzing the content of a computer system; specifically, we consider the examination of persistent storage, main memory, and network communications.

56.4.1 Storage Forensics

Persistent storage in the form of hard drives, optical disks, flash device, etc. is the primary source of evidence for most digital forensic investigations, in terms of both volume and importance. For example, during FY 2010, the average investigation at FBI’s Regional Computer Forensics Labs consisted of 470 GB of data [FBI10], and the vast majority of the data comes from hard disks.
Storage analysis, detailed by Carrier [Carr05], can be performed at several levels of abstraction:

- **Physical media**: At the lowest level, every storage device encodes a sequence of bits and it is, in principle, possible to use a custom mechanism to extract the data bit-by-bit. In practice, this is rarely done as it is an expensive and time-consuming process. The only notable exception is second-generation mobile phones for which it is feasible to physically remove (desolder) the memory chips and perform acquisition of content; Willasen [Will05] provides a detailed account of the process. Thus, the lowest level at which most practical examinations are performed is the host bus adapter (HBA) interface. Adapters implement a standard protocol (SATA, SCSI, etc.) through which they can be made to perform low-level operations. For damaged hard drives, it is often possible to perform at least partial forensic repair and data recovery [Moul07]. In all cases, the goal of the process is to obtain a copy of the data in the storage device for further analysis.

- **Block device**: The typical HBA presents a block device abstraction—the medium is presented as a sequence of fixed-size blocks, commonly of 512 or 4096 bytes, and the contents of each block can be read or written using block read/write commands. The media can be divided into partitions, or multiple media may be presented as a single logical entity (e.g., RAIDs). The typical data acquisition process works at the block device level to obtain a working copy of the forensic target—a process known as **imaging**—on which all further processing is performed.

- **File system**: The block device has no notion of files, directories, or even which blocks are considered “used” and which ones are “free”; it is the file system’s task to organize the block storage into a file-based storage in which applications can create files and directories with all of their relevant attributes—name, size, owner, timestamps, access permissions, and others. For that purpose, the file system maintains metadata in addition to the contents of user files.

- **Application**: User applications use the file system to store various artifacts that are of value to the end-user—documents, images, messages, etc. The operating system itself also uses the file system to store its own image—executable binaries, libraries, configuration and log files, registry entries—and to install applications.

Overall, analysis at the application level yields the most immediately useful results as it is most directly related to actions and communications initiated by humans. Indeed, most integrated forensics tools present a file system browser as their basic user interface abstraction. Yet, they also provide additional information that is not readily accessible through the normal operating system user interface. Such information is derived from additional knowledge of how a particular operating system works and what the side effects of its storage management techniques are.

One important source of additional data is the recovery of artifacts that have been deleted, or are otherwise not completely recoverable due to hardware errors. An extreme example of that is a used hard disk drive that has been freshly formatted—to the regular file browser interface, it would appear that the device is completely empty. In reality, only a small fraction of the drive has been overwritten—just enough to create blank file system metadata structures. The rest of the drive is marked as available but the previous content is still there, and that includes the complete content of almost all the files. In fact, the content would survive multiple formatting operations and, depending on usage patterns, data could be found years after its intended removal.

### 56.4.1.1 Target Acquisition

In line with best practices [Kent06], file system analysis is not carried out on a live system. Instead, the target machine is powered down, an exact bit-wise copy of the target disk is created, the original is stored in an evidence locker, and all forensic work is performed on the copy.

There are exceptions to this workflow in cases where it is not practical to shut down the target system and a live disk image from a running target is obtained; evidently, the consistency of data in use cannot be guaranteed. In virtual environments, a standard snapshot of the virtual disk can be trivially obtained.
The copy process of a physical disk can be accomplished in software, hardware, or combination of both. The original workhorse of forensic imaging has been the `dd` Unix/Linux general purpose command-line utility, which can produce a binary copy of any file, including special files representing entire storage devices. For example, the simple command `dd if=/dev/hda1 of=target.dd`, would produce a binary copy of the first partition of the first disk drive and will place it in the `target.dd` file. A hardware write blocker is often installed on the target device to eliminate the possibility of operator error that leads to the accidental modification of the target. Further, cryptographic hashes are computed for the entire image and (optionally) for every block. The latter can be used to demonstrate the integrity of the evidence if the original device suffers a partial failure, which makes it impossible to read its entire contents. To simplify the acquisition process, more specialized versions of `dd`, such as `dcfldd`, provide additional functionality, such as hashing and integrity verification on-the-fly, multiple output streams, and logging. A more advanced tool geared specifically toward forensics is the `guymager` target acquisition tool (Figure 56.2), which provides additional output options beside the raw image produced by `dd`.

`Ddrescue` is another variation on `dd`, which automatically tries to recover from read errors over multiple runs in an effort to extract as much data as possible from a failing hard drive. Virtually all commercial tools, as well as a number of other open source projects, provide the basic disk imaging capability, and it is beyond the scope of this discussion to provide an exhaustive enumeration. The National Institute of Justice sponsors the Computer Forensic Tool Testing (CFTT) project, which independently tests various basic tools, such as write blockers and image acquisition tools and regularly publishes reports on its findings.

### 56.4.1.2 File Carving and Reconstruction

Generally, data stores with dynamic allocation, such as file systems, main memory, and databases, contain leftover data that are not immediately accessible via the regular programming interface but are recoverable by direct examination of the raw data. For example, a file deleted by the user is simply marked as deleted and its storage allocation is reclaimed only when the need arises, which could happen anywhere from

---

* [http://www.cift.nist.gov/disk_imaging.htm](http://www.cift.nist.gov/disk_imaging.htm)
† [http://www.cift.nist.gov/hardware_write_block.htm](http://www.cift.nist.gov/hardware_write_block.htm)
¶ [https://savannah.gnu.org/projects/ddrescue/](https://savannah.gnu.org/projects/ddrescue/)
†† [http://www.cift.nist.gov/disk_imaging.htm](http://www.cift.nist.gov/disk_imaging.htm)
milliseconds to never. Main memory content can also persist for a long time as many computer systems are not reset for long periods of time, and even reboots may not clear memory [Hald08].

This behavior may be surprising to users, but it is primarily the result of performance considerations in the operation of software that manages the data. Sanitizing the de-allocated storage can be very expensive not only when applied to a hard disk but also to main memory management. This is a known security problem and solutions have been proposed [Chow05], but they are not routinely employed. Another reason to find data not actively in use in main memory is caching—perhaps the oldest performance optimization trick. Modern software systems use multiple levels of caching with the goal of minimizing secondary storage access, so data tends to persist where it was not intended to—main memory.

Forensic computing, unlike most other types of computations, is most keenly interested in all extractable artifacts, including—and sometimes especially—de-allocated ones. Unless a user has taken special measures to securely wipe a drive, it is reasonable to expect that, at any given time, the media contains recoverable applications artifacts—primarily files—that are ostensibly deleted.

File carving is the process of recovering application files directly from block storage without the benefit of file system metadata. The essential observation that makes this possible is twofold: (a) most file formats have specific beginning and end tags (a.k.a. header and footer); and (b) file systems heavily favor sequential file layout for better performance. We briefly survey the research and tool development in file carving; a more detailed look at the evolution of file carving techniques can be found in [Pal09].

In the simplest case, a file carving tool sequentially scans through a device image and looks for known headers at block boundaries. If a header is found, then the tool looks for a corresponding footer and, upon success, copies (carves out) all the data in between the two tags. For example, a JPEG image always starts with the header (in hexadecimal) FF DB FF and ends with FF DB. In practice, things are a bit more complicated since file formats were never designed for this type of processing and can have weak distinguishing features or may embed other file types (e.g., JPEGs can have an embedded thumbnail, also in JPEG). Further, actual data layout often has gaps, mixed-in blocks from other files, or outright data losses.

One of the first tools to implement the previously mentioned idea and gain popularity among forensic professionals was foremost.* It is a command-line utility which uses a configuration file for header/footer definitions and has some built-in parsing functions to handle Microsoft Office (OLE) files. Scalpel† [Rich05] is a file carver originally conceived as an improved implementation of foremost and utilizes a two-pass strategy to optimize work schedules and reduce the amount of redundant data being carved. Subsequent optimizations [Marz07] allowed the tool to utilize parallel CPU and GPU processing.

The development of new file carving techniques has been pushed forward by the 2006 [Carr06b] and 2007 [Carr07], DFRWS Challenges, which provided interesting scenarios for tool developers to consider. Garfinkel [Garf07a] performed a statistical study of actual used hard drives purchased on the secondary market and found that most fragmented files are split in two pieces and are, therefore, still recoverable with reasonable effort. He also presented a proof-of-concept approach based on using application-level verification.

Sencar and Memon [Senc09] developed a robust algorithm and a specialized tool that can carve out JPEG fragments and successfully reassembles them using image analysis techniques. The tool has been subsequently commercialized and can be useful to regular users to recover images from flash memory used in digital cameras.

Looking ahead, file carving is likely to be a useful tool for the foreseeable future as hard disk sizes continue to grow and it becomes ever more performance-critical for file systems to lay out files sequentially. At the same time, King and Vidas [King11] have established experimentally that file carving would only work in a narrow set of circumstances on modern solid state drives (SSD). The reason lies in the fact that

---

* http://foremost.sourceforge.net
† http://www.digitalforensicssolutions.com/Scalpel/
SSD blocks need to be written twice in order to be reused—the first write resets the state of the block, thereby enabling its reuse. To improve performance, the trim command was added to the SATA protocol specification to allow the operating system to garbage collect blocks not in use and make them ready for reuse. Virtualized cloud storage represents another limitation to file carving as de-allocated data are automatically sanitized to prevent data leaks across tenants.

### 56.4.1.3 File System Analysis

Once a working copy of the investigative target has been obtained, the actual analysis can commence in earnest. The first goal is to identify and gain access to all the artifacts stored on the media—files, directories, email messages and other personal communications, system registry entries, etc. The complete recovery of all useable artifacts requires deep understanding of the physical layout of the on-disk structure of the specific version of the specific operating system. This is a particularly thorny issue with proprietary file systems, such as the MS Windows family, as their structures are not publicly documented and need to be reverse-engineered. Fortunately, the open-source community at large has tackled many of these problems for interoperability reasons, and has developed some robust implementations.

It is useful to place the forensically significant file system data into two categories—system artifacts and application artifacts—which we discuss for the rest of this section.

**System artifacts**: The operating system continuously generates traces of its users’ activities. Some pieces of this information are quite obvious—automatic updates to timestamp file attributes—others are more obscure, such as system log entries. More subtle information can be obtained by observing the physical layout of the files on the block devices. In Unix/Linux, the structure containing the metadata for a file is called an *i*-node and each file is associated with a particular *i*-node. Normally, files created in a quick succession (during installation) would have the same creation timestamp and *i*-node numbers that are close to each other. However, if we find a file which blends in based on timestamp but has an uncorrelated *i*-node number (suggesting creation at a later time), this would be an outlier event worthy of further investigation.

We use *The Sleuthkit* (TSK), the most popular open source tool for file system forensics, to illustrate the system artifacts that are commonly used in a forensic investigation. TSK is C/C++ library, as well as a collection of command-line tools. TSK’s code is based on The Coroner’s Toolkit (TCT), which was originally created by Dan Farmer and Wietse Venema and featured in their classic book on forensics [Farm05]. The scope of TCT was expanded from the original target of Unix file systems to include MS Windows and MacOS formats.

TSK provides over 20 individual tools that can provide file system information at different levels of abstraction. Here, we provide a sampling of the more important ones.

**Metadata layer tools** process the on-disk metadata structures created by the file system to store standards attributes of a file—name, size, timestamps, etc. Under normal operating system operation, these structures are only available to the kernel. This toolset, however, allows the direct access to the following structures:

- `ifind` finds the metadata structure that has a given file name pointing to it or the metadata structure that points to a given data unit;
- `icat` extracts the data units of a file, which is specified by its metadata address;
- `ils` lists the metadata structures and their contents;
- `istat` displays the statistics and details about a given metadata structure.

Many modern file systems, such as ext3 and NTFS, employ a journaling (log) function, which keeps track of attempted file operation. Should the system crash, the journal allows for the replay of these operations and thus minimizes the loss of data to the user. TSK contains two tools—`jls` and `jcat`—that list the entries in the file system journal and can display the contents of a specific journal block, respectively. This enables an investigator to recover additional information that is not otherwise accessible.

* http://sleuthkit.org
The purpose of the block device tools is not just to provide access to individual data blocks but also to assign meaning to the content of the block. Specifically, we could target individual blocks that are allocated, unallocated, or we could identify and gain access to the slack space at the end of a block:

- **blkcat** extracts the contents of a given data unit (block);
- **blkls** lists the details about data units and can extract the unallocated space of the file system;
- **blkstat** displays the statistics about a given data unit in an easy-to-read format;
- **blkcalc** calculates where data in the unallocated space image (from blkls) exist in the original image. This is used when evidence is found in unallocated space.

**Volume system tools** take an image as input and analyze its partition structures. These can be used to find hidden data between partitions and to identify the file system offset for TSK:

- **mmls** displays the layout of a disk, including the unallocated spaces;
- **mmstat** displays details about a volume system (the type);
- **mmcat** extracts the contents of a specific volume.

As the previous descriptions suggest, in every group of tools, we can find a “stat” tool, which provides basic information (statistics) for the relevant data structures; an “ls” tool, which enumerates data structures of a particular kind; and “cat” tools, which interpret and extract the data from the structures.

The Autopsy Forensic Browser provides a simple Java-based graphical interface to TSK, as well as a framework that allows third-party components to be added to the processing pipeline. The goal of the project, still in its early stages, is to put together an open integrated environment that improves the usability of open source tools and enables further extension of functionality.

The registry is a central hierarchical database of key/value pairs used in all versions of Microsoft Windows to store information that is necessary to configure the system for one or more users, applications, and hardware devices. As such, the registry is a treasure trove for an investigator as it records a large number of values that can indicate user and application activity from recently used files, to traces of uninstalled applications, to records of external storage devices that have been connected to the system.

Tools like RegRipper* and RegistryDecoder† offer the option of extracting any available information from the registry. Since a registry contains tens/hundreds of thousands of entries, the investigator must have a fairly specific idea of what to look for. Figure 56.3 shows RegistryDecoder in action, where it has recovered the list of USB devices seen by the system and allows the investigator to tie the computer system to specific devices, including in this case a mobile phone.

Modern operating systems automatically create copies of user and system data in order to both improve reliability, by being able to reset to a known good state, and to support user data backup needs. In MS Windows, the service is called **Volume Shadow Copy Service‡** and captures stable, consistent images for backup on running systems under a variety of circumstances with regular users largely unaware of its operation. As a result, a forensic analyst can retrieve the prior state of the system, including artifacts that the user might have securely deleted [Harg08].

The file systems used by Unix derivatives—HFS+ on MacOS, ext3/ext4 on Linux, and ZFS on Solaris—use journaling as a basic reliability mechanism. By understanding the structure of the journal entries, it is possible to recover the recent history of file operations [Burg08, Fair12, Beeb09].

Application artifacts: An analyst can gain access to all the files created explicitly and implicitly by the user. Most users are well aware of the files they create explicitly (documents, images, emails), and a guilty user may try to cover their tracks by removing incriminating ones. Yet, computer systems tend to create multiple copies of the same information and can be very difficult to dispose them all. Modern applications contain numerous features to help the user by remembering preferences and histories, as well as

---

* [http://regripper.wordpress.com/regripper/](http://regripper.wordpress.com/regripper/)
† [http://www.digitalforensicssolutions.com/registrydecoder/](http://www.digitalforensicssolutions.com/registrydecoder/)
automation features such as recovery of previous document versions and automatic filling of author, date, title, and other properties. These could all be exposed and, together with the documented behavior of the application, could bear witness to what users did. It is fair to say that many applications are also sloppy during uninstall procedures and can leave (sometimes deliberately) traces behind.

Of particular interest are artifacts created as a side effect of the user’s web browsing activities. Browsers routinely retain a large amount of cached information about its users’ online life—history, searches, bookmarks, cookies, screenshots, open tabs, etc. As Altheide describes in detail [Alth11], most of these data are now stored in common data formats, which makes it trivial to recover and interpret.

Another rich source of information is the metadata that are routinely embedded in various artifacts, such as author information and version information in office documents, and camera and GPS information in image data.

Beverly et al. [Beve11] demonstrated that the average disk contains a significant number of IP and MAC addresses that can be extracted by means of carving and validation. In a sample of 723 hard drives, they found an average of 2279 IP addresses. In addition to the obvious use of connecting a drive to a network, the addresses could be used via geolocation services to map the physical location of a target machine.

Cohen et al. [Cohe11] provide a different perspective on forensics—one in which security monitoring, incident response, and forensic tool are put together in a large distributed system to keep Google’s expansive IT infrastructure secure. In building a distributed system to perform these tasks, a number of conflicting requirements, such as performance, privacy, and network connectivity, lead to technical solutions that would be considered incomplete in a classical forensic investigation.

56.4.2 Memory Forensics

The original view of best forensic practices (from a decade ago) was to literally pull the plug on a machine that is to be impounded. The rationale was that this removes any possibility to alert processes running on the host and would preempt any attempts to hide information. Over time, however, experience showed that such concerns were overblown and that the substantial and irreversible loss of important forensic information is simply not justified.
Memory analysis can be performed either in real-time on a live (running) system, or it could be performed on a snapshot (memory dump) of the state of the system. In live forensics, a trusted agent (process) designed to allow remote access over a secure channel is pre-installed on the system. The remote operator has full control over the monitored system and can take snapshots of specific processes, or the entire system. By and large, live investigations are an extension of regular security preventive mechanisms, which allows for maximum control and data acquisition. However, such mechanisms need to be pre-installed, which is only practical in commercial and government entities that can afford the expense of maintaining such a system.

In the general case, memory forensics reduces to forensics of a snapshot (a.k.a. dead forensics). It should be evident that analyzing a snapshot is considerably more difficult because we lack the run-time facilities and context provided by the live operating system. We need fundamentally new tools that can rebuild as much as possible of the state of the running system in a forensically sound manner, and, therefore, such tools are the main point of discussion for the rest of this section.

### 56.4.2.1 Memory Acquisition

Studies have illustrated that data persist for a long time in volatile memory ([Chow04], [Solo07]). Unlike tools that analyze running machines, off-line memory analysis tools extract digital evidence directly from physical memory dumps. These memory dumps may be acquired using a number of different mechanisms (dependent on operating system type and version), from hardware-based approaches, such as Tribble [Carr04] and via the Firewire interface [Bock], to software-only approaches [Vida06], such as using `dd` to access the physical memory device or via insertion of custom kernel modules. Vomel [Vome11] provides a comprehensive survey of modern acquisition techniques.

These memory dumping mechanisms are not infallible and some high-tech approaches to subverting memory acquisition have been proposed [Rutk07]. Fortunately, unless the subversion mechanism is very deeply embedded in the operating system, a substantial amount of overhead may be incurred to prevent acquisition, potentially revealing the presence of a malicious agent [Korn06a, Korn07]. Ruff and Suiche [Ruff07] developed a tool that provides another alternative for memory acquisition, by converting Windows hibernation files to usable memory dumps. Schatz proposed a novel approach to memory acquisition technique called BodySnatcher, which injects a small, forensic operating system that subverts the running operating system [Scha07]. Sylve's work [Sylv12] on Android memory acquisition* developed ostensibly for mobile devices works equally well on traditional Linux deployments.

Once obtained, a RAM capture contains a wealth of information about the run-time state of the system, with the most important listed in the following:

- **Process information**: We can enumerate and identify all running processes and threads and their parent/child relationships; we can enumerate all loaded systems modules, and can obtain a copy of the processes’ code, stack, heap, code, and data segments. All of this information is particularly useful in analyzing compromised machines as it allows us identify suspicious services, abnormal parent/child relationships, and to scan for known malicious code.
- **File information**: We can identify all open files, shared libraries, shared memory, and anonymously mapped memory. This is useful for identifying correlated user actions and file system activities and to prove more difficult implications, such as user intent.
- **Network connections**: We can identify open and recently closed network connections, protocol information, as well as send and receive queues of data not yet sent or delivered, respectively. This information could readily be used to identify related parties and communication patterns among them.
- **Artifacts and fragments**: Just like the file system, memory management systems tend to be reactive and leave a lot of artifact traces behind. This is primarily an effort to avoid any processing that is not absolutely necessary for the functioning of the system. As well, caching of disk and network data can leave traces in memory for a long time.

---

56.4.2.2 Memory Analysis

The 2005 DFRWS memory analysis challenge* has served as an early catalyst to push the development of tools for Windows memory forensics and resulted in several early projects.

**KnTTools**† extracts information about processes, threads, access tokens, the handle table, and other operating system structures from a Windows memory dump. The **memparser**‡ tool has the added ability to detect hidden objects by cross-referencing different kernel data structures. Schuster's **ptfinder** tools ([Schu06a], [Schu06b]) take a different approach: instead of walking operating system structures, they attempt to carve objects that represent threads and processes directly. Noting the relative dearth of usable tools for deep parsing of Linux memory captures, DFRWS created its 2008 challenge [Geig08] with a major emphasis on Linux memory forensics. Prior to that, the most notable work is Burdach's [Burd] proof-of-concept tool called **idetect**, which parses kernel 2.4-series memory dumps and enumerates page frames, discovers user mode processes, and provides detailed information about process descriptors. Case et al. [Case10] demonstrated the most advance memory parsing techniques, which allows their tool, **ramparser**, to find kernel structures even in custom Linux distributions, which are very common on various devices.

The most influential project in memory forensic has been the development of Volatility Framework§ led by Aaron Walters [Petr06, Walt07]. It provides core memory processing capabilities for Windows-based systems and a component architecture that allows third-party expansion. Volatility supports multiple input formats (raw, hibernation file, crash dump), and can extract running processes, loaded module, open network connections, open registry items, reconstruct executable binaries, and automatically map virtual to physical addresses. The project has become the focal point for most recent research and development efforts in memory forensics and has the potential to bring all practical techniques into a single package.

56.4.3 Network Forensics

Network forensics uses specialized methods to collect, record, and analyze transmitted data for the purposes of detecting or reconstructing a sequence of relevant events. Obviously, the main source of information is network traffic, which may be monitored in real time, or recorded and analyzed later. From the point of view of forensic reconstruction, there is no fundamental difference between the two cases and the same methods could be applied (subject to performance constraints). Most network traffic analysis is preventive monitoring and aims to identify and neutralize security threats in real time—firewalls and intrusion detection systems observe and actively manipulate network flows to achieve the desired security posture.

It can be useful to distinguish two types of network inquiries—incident response and forensic reconstruction—although they often blend together. Incident response teams deal with identified security breaches, and they function much like an emergency room team—quickly assess the immediate damage, stop the bleeding, and commence a recovery operation. Forensic reconstruction is a deeper analytical process to put together the sequence of events that lead to the incident, assess the wider scope of damages—which systems were compromised and what data were compromised—and try to attribute blame to the ultimate source of the attack.

56.4.3.1 Capture and Reassembly

Among general-purpose network capture and analysis tools, Wireshark¶ (a.k.a. Ethereal before 2006) stands out as the most popular and complete one in the public domain. It can analyze both live

---

* http://dfrws.org/2005/challenge/
† http://www.gmgsystemsinc.com/kntools/
‡ http://sourceforge.net/projects/memparser
§ https://www.volatilesystems.com/default/volatility
¶ http://wireshark.org

© 2014 by Taylor & Francis Group, LLC
traffic and network captures. Its main interface window is shown in Figure 56.4. It consists of three components—a table of captured network packets, a detail panel showing the interpretation of the selected packet, and a raw hexadecimal/text display at the very bottom. Since the number of packets quickly becomes overwhelming, the tool provides a filtering facility, which allows for selection based on any of the fields shown—source, destination, protocol, content, etc.

Recall that the purpose of the Internet protocol stack is to provide end-to-end communication channels for distributed processes that execute on connected hosts. The data sent by processes are transformed by four different layers—application, transport, network, and data link—each of which leaves its footprint on the observed traffic. In particular, the flow of data sent is split into protocol data units and each layer adds protocol-specific header to the original message. During transmission, pieces from different connections are interleaved, which leads to the need to reconstruct the entire conversations to make sense of them; Figure 56.5 shows an example of a reconstructed conversation between a web client and server.

56.4.3.2 Analysis

Network data analysis follows the earlier mentioned process and can be performed at the four different layers of abstraction, depending on the focus of the investigation. At the transport layer, the most important pieces of information added in the header are the source and destination port numbers, which locally identify the communicating processes at the sender and the receiver, respectively. The network layer header contains, among other things, the source and destination IP addresses, which identify the sending and receiving hosts. Finally, at the data link layer, the header contains the physical (MAC) addresses of the source and destination for the current hop on the LAN. For every hop, that information will be different as different hosts, using potentially different link types (Ethernet, 802.11, FDDI) will be forwarding the data.

Thus, if a case revolves around a wireless network breach, the most relevant data are likely to be found at the data link layer. If the issue is a botnet attack coming from the Internet, then most of the relevant data will probably be in the network and transport layers. If the problem is data exfiltration, the best starting point is likely to be the application layer, as a user is likely to use a ready network application, like email, to transmit the data.

Automating the process of network forensics is an ongoing challenge; the sheer volume of data means that, in order to find the artifacts of interest, one needs to know rather specifically what to look for.
Most network monitoring tools, such as NetWitness,* provide rule-based filtering and aggregation and can alert an analyst that something is amiss. For example, an increase in malformed packets, or abnormal levels of traffic could be an indication of an attack attempt, as would be connections on unusual for the advertised protocol port, connection attempts to known bad hosts, obfuscated content, etc.

56.5 Artifact Analysis

Another approach to understand forensic processing is to consider different methods for processing digital artifacts. We can take advantage of the fact that the representation of many artifacts—text, images, audio, video—remains consistent across different media and devices and we could develop generic methods that could be applied across the board.

56.5.1 Crypto Hashing

The lowest common denominator for all digital artifacts is to consider them a sequence of bits/bytes without trying to parse them, or assign any meaning. Despite this low level of abstraction, there are some very important problems that can be addressed and the most important one of them is to find known content.

Hashing is the first tool of choice in investigating any case, as it provides the reliable means to validate data integrity and to identify known content. At a basic level, hash-based methods are attractive due to their high throughput and memory efficiency. Recall that a hash function takes an arbitrary string of binary data and produces a number, often referred to as digest, in a predefined range. Ideally, given a set of different inputs, the hash function will map them to different outputs.

Hash functions are collision-resistant if it is computationally infeasible to find two different inputs for which the output is the same. Cryptographic hash functions, such as MD5, RIPEMD-160, SHA-1, SHA-256, and SHA-512, are explicitly designed to be collision-resistant and to produce large, 128- to 512-bit results.

* http://www.netwitness.com/
Since the probability that two different data objects will produce the same digest by chance is astronomically small, we can assume that, if two objects have the same digest, then the objects themselves are identical.

The current state-of-the-practice is to apply a cryptographic hash function, typically MD5 or SHA-1, either to the entire target (drive, partition, etc.) or to individual files. The former approach is used to validate the integrity of the forensic target by comparing before-and-after results at important points in the investigation. The latter method is used to eliminate known files, such as operating system and application installations, or to identify known files of interest, such as illegal ones. The National Institute of Standards and Technology (NIST) maintains the National Software Reference Library,* which covers the most common operating system installation and application packages. Similarly, commercial vendors of digital forensic tools provide additional hash sets of other known data.

From a performance perspective, hash-based file filtering is very attractive—using a 20-byte SHA-1 hash, we could represent 50 million files in 1 GB. Thus, we could easily load a reference set of that size in main memory and filter out, on the fly, any known files in the set as we read the data from a forensic target. In addition to whole files, we are often interested in discovering file remnants, such as the ones produced when a file is marked as deleted and subsequently partially overwritten.

One routinely used method to address this problem is to increase the granularity of the hashes—we can split up the files into fixed-size blocks and remember the hashes for each individual block. Once we have a block-based reference set, we could view a forensic target as merely a sequence of blocks that can be read sequentially, hashed, and compared to the reference set. Typically, the block size is 4 KB to match the minimum allocation unit used by most operating systems installations. There are two main advantages to this scheme—we can easily identify pieces of known files and avoid reading the target hard drive on a file-by-file basis [Garf10].

**56.5.2 Approximate Matching**

So far we looked at ways to find artifacts that are identical; however, we could consider the more general problem of finding objects that have similar binary representation. Inspired by earlier work in spam filtering, Kornblum [Korn06b] proposed a fuzzy hash construction. It uses shingling to split up the file into chunks; generates small, 6-bit hashes for every chunk; and concatenates them to produce the final hash, which is base64-encoded. This scheme is a derivative of Rabin's groundbreaking work on data fingerprinting by random polynomials [Rabi81]. For the similarity comparison, the two hashes are treated as text strings and are compared using an edit distance measure, which produces a number between 0 and 100.

An interesting design choice is to limit the overall size of the hash to 80 symbols—this is mainly driven by the aim to give investigators a fixed hash value per file similar to the ones produced by crypto hashes (MD5, SHA-1, etc.). To achieve this, the algorithm uses the file length to estimate the number of piece it needs to break the file into. After the hash calculation is completed, if the result is longer than the target 80 symbols, the estimate is doubled and the calculation is redone from scratch. Due to its sensitivity to object size, the actual file signature produces two hashes for two different resolutions—c and 2c. This takes the edge off the problem; but if the difference in size between the data objects exceeds a factor of four, the two are not comparable. In practice, the hash does seem to work well in identifying objects that are versions of each other and are not too big and not too dissimilar; however, the hash quickly loses resolution for larger files and cannot be applied to stream data where size is unknown.

Roussev et al. [Rous06] proposed a similarity hash measure, primarily targeted at detecting versions of executable files and libraries. The essential idea is to break up the object into known constituent parts (coded functions and resources), hash each one individually, and combine them into a Bloom filter to produce the similarity hash. Hashes are then compared by counting the number of corresponding

* http://www.nsrl.nist.gov

© 2014 by Taylor & Francis Group, LLC
bits in common between the two filters and comparing them with the theoretical expectations—any statistically significant deviation is an indication of similarity. Performance results for system libraries demonstrate that versions are readily detectable by this method, even when filters with very high compression rates are used.

The follow-up work on multi-resolution similarity (MRS) hashing [Rous07] addressed the problem of comparing objects with vastly different size, such as a file and an entire drive. To balance the requirements of accuracy and performance, it was proposed to generate similarity hashes at multiple levels of resolution and to choose an appropriate level, depending on the specific objects being compared.

In [Rous10] Roussev proposed a new approach to generating similarity data fingerprints based on the idea of statistically improbable features. In essence, from every neighborhood the scheme selects a few 64-byte features (byte sequences) that, based on a statistical survey, are least likely to appear by chance. The method relies on the calculation of the Shannon entropy of each sequence and has the added advantage that low-entropy features likely to trigger false positive results are eliminated. This is a different approach from the random polynomials, and the user study in [Rous11] clearly demonstrates that: (a) the approach performs better than random polynomial as low-entropy data are eliminated from consideration; (b) the bit stream similarities found almost always correspond to user observable similarity. Overall precision and recall rates were in the 95% range. Subsequent work further developed the concept to allow parallel computation of the digests [Rous12a] and applied similarity digests as a primary triage technique to a 1.5 TB case study [Rous12b] with over 120 different data sources.

56.5.3 Multimedia

The ability to attribute a multimedia artifact to its source, as well as the ability to identify forged artifacts, is of critical importance to a forensic investigation. Our discussion is focused on images, but similar concerns and methods apply to audio and video artifacts. The problem matching an image and its camera source has been approached from two perspectives: (1) identify the type of camera that has taken this photograph; or (2) prove that the image has been produced by a specific camera.

In [Fang09], the researchers introduce a forensic scheme to distinguish between DSLR and compact images. Since DSLR and compact cameras use different type of sensors and lenses, their camera output quality in terms of sharpness and ISO sensitivity differs significantly. Such differences also affect the sensor noise levels and can be detected through wavelet decomposition and noise analysis. The experimental results show that the proposed forensic scheme has a potential to identify DSLR and compact images even when they are recompressed or down-sampled by 50%.

Lukas et al. [Luka06] proposed a method for identifying the source camera of a digital image based on the imaging sensor’s noise pattern. The pattern is extracted by considering a large sample of the images from the camera, and the attribution of images is performed through statistical correlation. To improve the applicability of the method as a forensic tool, Sutcu et al. [Sutc07] proposed an enhancement based on artifacts produced by the demosaicing operation (color interpolation performed by the color filter array of the camera).

Sophisticated image manipulation has become a commodity technology, and it is becoming near impossible to spot forgeries simply by looking at the image. Farid has an excellent survey [Fari09] on the various techniques that could be applied to detect forgery. Here, we offer a brief summary of the different classes of techniques.

Pixel-based techniques work by analyzing the image to detect the most common types of image tampering like cloning, resampling, and splicing. In all cases, the basic observation is that, although the image is perceptually sound, the underlying transformation creates detectable artifacts that can be exposed and detected via statistical classification.

© 2014 by Taylor & Francis Group, LLC
Digital Forensics

Format-based technique works by considering the effects of lossy image compression, such as the popular JPEG, on the image. Namely, the image is encoded based on a lattice of 8×8 blocks, and the DCT coefficients of the block are quantized and encoded. This process is the main source of loss and produces (not always visible) artifacts along the boundaries. These can be used to detect various forms of tampering.

Camera-based techniques can identify the fact that different pieces of the image do not come from the same camera and this could be established using the attribution techniques outlined earlier. Another technique to spot manipulated images is to consider lighting—it is quite difficult to have two photographs taken under the exactly same conditions. By analyzing the direction from which the light shines on different objects [John 05], it is practical to determine that the image is a composite. For human subjects, the eyes provide a perfect point of reference as their high reflectivity often provides a clear indication of the positions of light sources.

Geometry provides another approach for detecting image tampering. In authentic images, the projection of the camera center onto the image plane is near the center of the image. When a person or object is translated in the image, the principal point is moved proportionally, so any differences in the estimated principal point across the image can be used to detect tampering. In [John07], Johnson and Farid apply this idea of using a pair of eyes, or other suitable objects, to estimate the camera’s principal point from the image.

56.5.4 Event/Timeline Reconstruction

Timelines are one of the most frequently used tools in forensic analysis. Typical data sources, like the file system, provide generous amounts of timing information as every creation, access, or modification to a file leads to update of their MAC (modification/access/creation) timestamp attributes. The three main problems related to timelines are provenance, synchronization, and visualization.

The problem of establishing the provenance and reliability of timing information found on a target stems from the fact that timestamps carry no special protection and could be manipulated by users. Users with root access can modify the system clock or manipulate any timestamp directly. Even if no tampering is done, establishing the order of events taken place on the Internet is not as easy as it may appear—no two hardware clocks advance at the same rate, which means that they tend to drift away from real-world time and from each other.

Most hosts use NTP to periodically synchronize with time servers and, thereby, limit the amount of drift. Yet, the error is always present and is complicated by misconfigured clients and servers. Time zones, differences in time-keeping methods on different platforms, and “clock” events like daylight savings time, which are applied very unevenly around the world, further complicate efforts to establish the order and timing of events. In 2007, Buchholz [Buch07] performed a sizeable survey of computer clocks on the Internet and documented these pathologies using a large amount of empirical data.

Forensic timeline visualization tools have considered different approaches to aggregating the timing information—from lower to higher level event and displaying it to the examiner. On the left, Figure 56.6 shows the approach taken by Zeitline [Buch05]—one of the early systems. The essential idea is that investigators can go over the ordered list of events and create different timelines related to different aspects of the investigation. They can also group low-level events and annotate them as higher-level events, ultimately building a hierarchy out of the flat timing information.

TimeLab [Olss09] takes a more automated approach—it uses a number of specialized scanners to process different sources of timing information both from artifact metadata and from system logs. It displays the different timelines next to each other and has basic facilities to zoom in and out to refine granularity of the display.

The log2timeline tool [Gu10] is the most comprehensive timeline extraction tool used by practitioners—it parses everything from file time stamps and event logs to registry entries and browser artifacts. It can quickly produce a large amount of data, even for a modest data source. The challenge for the investigator
is soft through, which is typically performed by custom scripts. The visualization is treated as a separate concern and can be performed with general timeline tools like SIMILE Timeline widget* to provide a more intuitive navigation (Figure 56.7) and exploration.

56.5.5 Application Forensics

Application forensics refers to the process of reliably establishing the behavior of applications as the result of user actions, and the identification of associated artifacts. Once such causal links are established, we can look for the artifacts to infer user behavior. As the name suggests, application forensics is inherently tied to the piece of software being used, and it is difficult to generalize this work across applications.

One prominent example of frequently used application forensic analysis is peer-to-peer (P2P) networks. Despite its numerous legitimate uses, P2P has also gained notoriety as one of the major means of distributed contraband material, as well as a major source of copyright infringement complaints. At first glance, one could be tempted to classify this as a network forensics problem. However, a closer look at the methods used

* http://www.simile-widgets.org/timeline/
reveals that no actual analysis of network traffic is performed. Since the purpose of P2P is file sharing, it is much easier to handle the problem on the end system by examining the artifacts from the P2P interactions.

*P2P Marshal* [Adel09] is a tool which automatically detects and analyzes P2P client use found on a particular target machine. The tool aims to automate end-to-end the process of investigating a target for all known P2P activity. It determines what clients are currently installed (or have been installed) on a machine, and then extracts per-user usage information for each client, including lists of shared or downloaded files and peer-servers contacted. Throughout the process, the tool performs its actions in a forensically sound way and maintains a detailed log of all actions performed.

Liberatore et al. [Libe10] provide a more general technical and legal context for investigating P2P networks. Specifically, they study the problem of identifying and tracking individual users in Gnutella and BitTorrent and show the problems arising from hearsay information distributed through the networks.

### 56.6 Forensic Tool Design and Implementation

From a historical perspective, computer forensics tools are relatively young in their development cycle. At this stage, several commercial vendors have established strong presence in forensic labs by providing integrated investigative environments that attempt to support the complete inquiry cycle from acquisition to reporting. Not surprisingly, these vendors would like to retain their competitive advantage. To achieve this, they have created a “walled garden” environment, trying to lock in clients and to establish their standards as the de facto standards for the community.

Much of the advanced research in forensics is produced in academia, which simply lacks the incentives and the resources to develop fully functional integrated environments. Instead, researchers tend to focus their effort on a specific problem and to produce a narrowly targeted tool, which often requires climbing a learning curve before it can be put into practical use. Based on the experience of more mature software domains, the above situation is typical—commercial tools in the field are ready for practical use, but rarely employ the most advanced techniques.

The grand challenge for forensic tool design is to build tools that closely match the cognitive task of forensic analysis. Current generation tools are squarely focused on the information foraging loop (Figure 56.1). Specifically, they emphasize the gathering part of the loop, with most of triage and filtering functions primarily performed by the analyst.

#### 56.6.1 Scalability and Automation

The need for scalable distributed forensic processing was first identified by Roussev and Richard [Rous04, Rich06] as a critical issue that forensic tools will face. Indeed, increasing data volume is already taxing existing systems to the point where slow performance is becoming one of the top concerns for investigators [Hibs11], along with the need for smarter and more automated tools. Coincidentally, more automation and smarter tools also imply more processing, specifically more compute-intensive processing. Thus, building scalable tools that can handle large cases is a problem of crucial necessity.

The efforts so far have been relatively few despite the early work in [Rous04], which showed that even a modest research prototype can speed processing considerably; in some cases, super-linear speed up has been observed due to cumulative caching benefits. In [Rous09], another research prototype showed the suitability of Map-Reduce style for forensic processing by quantifying its performance on a small cluster using a custom map-reduce system.

#### 56.6.2 Standards and Inter-Operability

Given the wide range of possible forensic analysis tasks, it is difficult to conceive that any particular integrated product would cover everything. Therefore, it is in the interest of the community at large to develop standards that allow for interoperability among tools.
The push for standards has been around for a number of years and has had some high-profile backers, such as the National Institute of Justice, which supported the development of the Digital Evidence Markup Language (DEML). It appears, however, that this effort to define a comprehensive standardized description standard has gained little traction and has all but disappeared into obscurity.

In the real world of forensics, the most pressing concern is the storage of real evidence sources. For some time, the de facto standard for digital evidence containers has been the Expert Witness Format (EWF) developed and maintained by Guidance Software—the maker of the EnCase Forensic product. This is a proprietary format, which has been thoroughly researched and reverse-engineered by the makers of libewf*—an open source library for manipulating EWF containers.

A strong, open alternative has been the afflib project,† which produced the Advanced Forensic Format (AFF) [Garf, Garf06] that, from a purely technical design perspective, is more advanced and forward looking than EWF [Cohe09]. A number of efforts have centered on the idea of using XML containers for digital evidence, most notably XIRAF [Alin06] developed the Netherlands Forensic Institute, and DEX [Levi09] from the University of Massachusetts. The main drawback of using XML is that it is quite verbose and inefficient for the large amounts of data that need to be stored for every case. To become a practical alternative, an XML-based scheme must incorporate compressed storage without compromising the inherent query advantages of XML containers—a non-trivial design challenge.

With the recent push toward the use of memory forensics methods, the need has emerged to define a suitable container format for imaged memory targets. Schuster [Schu08] has commenced an effort to adopt an early standard for such evidence.

The most comprehensive and mature research and development effort is the courtesy of Simson Garfinkel [Garf12]. The Digital Forensics XML (DFXML) is an XML language that enables the exchange of structured forensic information. DFXML provides the means to comprehensively describe any piece of data relevant to an investigation—volumes, files, registry entries, etc.—and maintains provenance information.

56.6.3 Tool Testing and Calibration

One fundamental difference between forensic tools and most other tools is that the legal system and the society at large must be able to trust the results of a forensic inquiry. For that purpose, each tool should be tested and its performance and error rate thoroughly documented. In practice, only simple tools, such as data cloning ones, have been independently tested and this remains a major challenge for researchers and developers.

One of the most important research efforts is the establishment of representative public data sets that can be used for research and testing. This requires a significant effort in any discipline, but forensics presents at least three added challenges: (1) the actual data on which investigations are performed cannot be published; (2) each case is different, so establishing a representative set requires a huge amount of data gathering; (3) we have no credible methods of generating simulated data, especially for complex scenarios.

The DFRWS conference (dfrws.org), through its annual challenges, has contributed several data sets that are often used as benchmarks in evaluating file carving and memory analysis tools. Garfinkel et al. [Garf09] have collected and published several important data sets, such as the real data corpus (U.S. and non-U.S. sets), which consists of used drives obtained on the secondary market (6.7 TB), the govdocs corpus (1 TB), which consists of a million copyright-free U.S. government documents, and the M57 corpora (1.5 TB), which consists of the data collected during a scripted exercise on a private network.

* http://www.sleuthkit.org/informer/sleuthkit-informer-23.txt
† http://afflib.org/
56.7 Summary

In this chapter we presented a brief overview of the state of the art and state of the practice in digital forensics. Within the constraints of a single chapter, the primary goal has been to provide broad conceptual coverage of this growing research field as opposed to in-depth discussion of specifics.

As a computer science discipline, digital forensics is a smaller and younger field relative to the broader area of security and privacy. Its initial development was somewhat ad hoc as it was driven by the daily needs of forensic examiners who did not necessarily have in-depth technical knowledge and had to learn on the job. Over the past decade, however, a growing number of computer science researchers have entered the field and have made progress in establishing more rigorous scientific approach to the development and testing of digital forensic methods.

Researchers and practitioners have developed a large number of specialized methods to identify, extract, and forensically analyze digital artifacts that are resident in main memory, stored on secondary storage, or transmitted over the network.

The foreseeable future promises to continue to bring exponential growth in forensic data, and increasingly complex processing due to the growing number of applications and devices, and their interactions. Increasingly, the type of deep analysis where a human investigator sifts manually through large amounts of data will become infeasible.

56.8 Open Issues and Challenges

Looking forward, digital forensics faces new challenges as new technologies continue to enter our lives. Every new technology presents new challenges and increases the complexity of the forensic analysis, and an exhaustive list of upcoming challenges could be quite long. Therefore, we chose the ones we consider the most fundamental.

56.8.1 Scalability and Automation

Looking forward, the biggest challenge looming over forensics is the ever growing mountain of data that our society produces. FBI statistics show that between 2003 and 2010, the average amount of data per case has grown from 80 to 470 GB—an average annual growth rate of 28%. Such growth, along with technology trends in computing, storage, and communication, renders current single-machine approaches to investigation unsustainable. Forensic computing needs to be able to employ as many resources as necessary to keep pace with the needs of society. To answer the challenge, researchers need to develop new methods that allow for effective automated data reduction, massive parallel processing, and efficient collaboration among investigators.

56.8.2 Sensor and Device Diversity

Another important direction in technology development is the proliferation of various sensors and devices, which drive us toward a pervasive computing environment. Even defining the scope of an examination will become increasingly more difficult as the interconnections among the components open up a large number of potential lines of inquiry. Our current abilities to quickly investigate new devices are quite modest and predominantly focused on mobile phones.

56.8.3 Virtualization and Cloud Computing

Existing methods and procedures are closely tied to the hardware of the investigated system. It is clear that virtualization technology, which decouples a computation from its hardware platform, has become a commodity and will become pervasive in the next few years. Cloud computing is a natural progression
Managing and Securing the IT Infrastructure and Systems

of the virtualization idea, and it allows the computation to move the place in the network where it is most efficient to execute.

This is a fundamental change as it negates one of the fundamental assumptions in forensics—that a process is tied to its hardware. Other important consequences include the fact that virtualization means sharing of the same of same physical resources—memory, storage—by numerous, unrelated computations. This forces the infrastructure to sanitize all resources after use and eliminates a vast amount of evidence currently in use.

56.8.4 Pervasive Encryption

Expanding on the previous point, for a long time we have relied on fact that data on most media are readily accessible—encryption was always an option, but the cost was generally too high for most applications. The default state of affairs is about to change—encrypted data is becoming the norm with the introduction of hardware encryption on hard drives. Due to privacy concerns on cloud computing, it is not difficult to envision encryption becoming the norm for in-memory data.

So far, forensic investigators have relied on implementation flaws to get around the encryption problem; however, sooner rather than later, encryption implementations will be cleaned up and the data would be truly inaccessible. Indeed, proper use of encryption should make it inaccessible and there are powerful economic and societal interests that will push for getting it right.

56.8.5 Legal Issues and Proactive Forensics

As better security and privacy mechanisms are deployed ubiquitously, it is increasingly likely that a number of staple forensic techniques would drop in significance. In particular, it is reasonable to expect that operating systems and applications will be more careful about “shredding” de-allocated objects, thereby eliminating traces upon which we currently rely. The trend toward more network-centric cloud computing means that a number of legal issues, such as location of the data, can further reduce the available forensic data.

This leads us to the conclusion that more and more forensically relevant data would have to be collected pro-actively, as already suggested by Shields [Shie11] and as already practiced by many large organizations as part of routine monitoring. Further expansion of this data collection will likely require a proper legal framework. What shape this may take is beyond the scope of this chapter but it is fair to say that, currently, the legal system lags considerably behind the pace of technology development and has been in no rush to catch up.

Glossary

Cryptographic hash: A deterministic algorithm that takes an arbitrary block of data and returns a fixed-size bit string, such that any change to the data will change the hash value.

Digital/electronic evidence: Probative information stored or transmitted in digital form that can be used in legal proceedings.

File/data carving: The process of extracting and reconstructing digital artifacts from constituent pieces.

Forensic timeline: A sequence of user- and system-initiated events deduced from available time information from sources like logs, file metadata, and file system metadata.

Live forensics: The application of forensic methods to examine the state of a running system.

Memory forensics: The process of reconstructing the live state of the system from a snapshot of RAM contents.

Network forensics: The process of reconstructing the history and content of communication between devices of interest.
**Provenance of digital evidence:** An explicit description of the set of tools and transformations that led from acquired raw data to the resulting conclusion.

**Secure deletion:** The process of overwriting de-allocated artifacts such that subsequent reconstruction is not possible.

**Similarity/fuzzy hash:** A hash function that maps syntactically similar artifact to numerical values that are close to each other.

**References**


© 2014 by Taylor & Francis Group, LLC


© 2014 by Taylor & Francis Group, LLC
Managing and Securing the IT Infrastructure and Systems


[Rous10] Roussev, V. Data fingerprinting with similarity digests, *Sixth Annual IFIP WG 11.9 International Conference on Digital Forensics*, Hong Kong, China, January 2010.


[Rutk07] Rutkowska, J. Beyond the CPU: Defeating hardware based RAM acquisition tools (part I: AMD case), *Black Hat DC 2007* presentation.


