

# Live Forensics: Diagnosing Your System Without Killing It First

Frank Adelstein

**Abstract:** Traditional methods of digital forensics analyze a static disk image—a bitstream copy of a disk created while the system is offline. Recent trends—including greatly increased disk capacity and the proliferation of mission-critical systems requiring continuous uptime—have limited the effectiveness and applicability of this approach. *Live forensics* gathers data from running systems, providing additional contextual information that is not available in a disk-only forensic analysis. This article describes what information live forensics can gather, how to use that information as evidence, and what information is best obtained by live forensic analysis.

## Introduction

Fifteen years ago, servers for university computer science departments used disks with a two gigabyte capacity, and one megabyte floppy disks were still useful. Many computers were attached to the Internet, many of them malicious, many at low, dial-up speeds of 9.6Kbps. But, thanks to Moore's Law, that scale of operation is now considered tiny. Today's desktop computers commonly have two gigabytes of memory, and more than 100 gig of disk. A ten-person company can have over a terabyte of disk space on its desktop machines alone. Home computer users connect to the Internet with high-speed links of 3-6 Mbps, rivaling the bandwidth of smaller computer science department networks not long ago.

The nature of digital forensic investigation has therefore changed. Larger disk capacities increase the time required for analysis and the difficulty and expense of collecting all disk evidence. The pervasive nature of the Internet makes contextual information more important—specifically, who is connecting to a machine and what they are doing.

## Traditional forensics

In the traditional “snatch and grab” approach to computer forensics, an investigator pulls the plug on the machine, and then images (copies) the disk, either on site or (after confiscating the machine) in a lab. An analyst examines the image (actually a copy of the image) in a controlled environment—by repeatable steps on whose results both prosecution and defense will agree.

This approach has several drawbacks. First, it may not be possible. As systems continue to increase in size, terabytes of disk data are no longer uncommon and imaging can take many hours. Imaging is not possible, or at least extremely difficult on NAS, SANs and large RAID arrays. The time and effort needed for analysis increases with the disk size. A person can quickly peruse all the sectors on a one megabyte floppy disk.

Manual scrutiny of a 1G disk is not feasible; automated analysis assistance is required. Even automated analysis of a terabyte of data is slow [Roussev 2004].

While a disk is being imaged, it is offline, but for many systems—e.g., electronic commerce systems—the loss of revenue from even a few hours of downtime is unacceptable. Because of that, many judges no longer issue court orders to take down servers.

And finally, much information about what is happening on a running system is lost when the plug is pulled. This information provides the *context* for the disk evidence.

Traditional digital forensics attempts to preserve *all* (disk) evidence in an unchanging state, while live digital forensic techniques seek to take a snapshot of the state of the computer, similar to a photograph of the scene of the crime. For these reasons and more, interest in, as well as a need for, conducting a forensic analysis of live systems has been growing.

A warning and disclaimer: It is very easy to contaminate the evidence on a system. The many subtle points and quirks of various operating systems, as well as all of the rules, constraints, and limitations imposed by the applicable legal system are beyond the scope of this article. Therefore any forensic investigation should be conducted by a professional.

## **What information is available?**

Information available from a live system provides a *context* for the disk data—e.g., running processes, network connections, memory (process and physical), and other state items such as caches, logged on users, and system load. Live analysis can capture both this volatile information, and static information about the file system. Currently, most forensic tools use the operating system itself to obtain this information. If a machine has been compromised, its kernel can be malicious running code (a rootkit) that prevents the operating system from reporting the existence of processes and files. While there are programs that can detect the presence of rootkits (such as chkrootkit from [www.chkrootkit.org](http://www.chkrootkit.org)), little can be done if a rootkit is present except to fall back to the traditional approach.

A big concern in live digital forensics is that the system is not static—files and processes are continually changing. However, this does not necessarily invalidate them as evidence. For example, even though the system log files will continually change and new mail continually arrives, this activity will not *create* an incriminating mail message sent by the suspect days before the system is imaged. In addition, disk buffers may not have been written to disk due to caching. In other words, the context of the evidence is

significant. The time stamps associated with the files, called “MAC time” (modify, access, and “create” times)<sup>1</sup>, can help establish this time context.

## Acquiring the evidence

An investigation is an iterative process: repeatedly acquiring and analyzing data until a decision can be reached [Adelstein 2002]. The investigator acquires the volatile state of the running machine by running programs on it. Because gathering evidence on the target can affect *other* evidence on the target, a set of best practices has evolved to maximize the quality of the evidence. The most important are: running known good binaries, hashing all evidence, and gathering data in the order of volatility.

*Running known good binaries:* An investigator should not trust the executables on the running system, but should provide all of the executables used for gathering evidence. The executables should be statically compiled, if possible. Otherwise, they should include any shared libraries required by the executable. The programs should originate from a read-only medium, such as a CD-ROM. The executables can be copied to the running system; however this action will affect the disk, possibly overwriting evidence residing in deleted files. If the choice is between losing some evidence by overwriting and losing *all* evidence by not obtaining the information, it is better to risk the minor damage from copying files to the target system.

*Hashing all evidence:* Once acquired, evidence must be preserved in a way such that the investigator can later demonstrate that nothing has changed. The accepted method is to compute a cryptographically secure hash of the data (typically via MD5 or SHA-1). The hash represents a “fingerprint” of the data with a small number of bytes, typically 16–20. The hash can be recalculated later and compared with the original to show that the data has not changed from the time the original hash was obtained. If data is transmitted over a network from the target to another machine, the hash should be computed on both machines and compared to ensure no data changed in transit. Both the hashes and the evidence should be maintained in a secure location. Typically, an investigator preserves the integrity of the hash itself by signing and dating a printout of the hash and storing this in a secure location.

*Gathering data in order of volatility:* Some data are more ephemeral than others. Evidence should be gathered based on the Order of Volatility [Farmer 2004]. For example, open network connections change more frequently than the system load average or the users logged on to the system. Some actions may affect other data. For example, logging in to a system may generate entries in the system log files. Complicating matters is that the time required to gather evidence may depend on the kind of evidence gathered. A dump of the physical memory of a machine may be useful and is very volatile, indicating it should be accomplished early in the investigation. However, it can take tens

---

<sup>1</sup> The C time is sometimes referred to as the (inode) “change” time on Unix operating systems.

of minutes to complete, and during that time, more useful information such as the lists of running processes, open files, and network connections, will have changed or disappeared. And while overall the system RAM is continuously changing, on a modern system with a gigabyte or more of memory, many memory pages may linger for a considerable time (days or weeks). In other words, the investigator must be aware of the overall context of the investigation in order to make informed decisions on the order of evidence acquisition.

## **Analysis—putting it all together**

Once evidence has been acquired, it must be analyzed. In traditional forensics, the analyst gathers all potential evidence prior to analysis. It is often impractical to gather all possible information available to a live forensic examination. An investigator may therefore perform a *triage* and gather the essential data, examine it, and use the results of the first look to decide what else is needed. So for live forensics, the analysis step may lead to further acquisition of data. A live analysis creates opportunities for faster response. Consider an example:

The investigator receives a report of a slow web server. He obtains a list of running processes and the network ports they have open. One process has a connection to a high port number on an unknown system. The process also has a file open for writing. The file is a log of all network traffic, which indicates that the process creating it is a network sniffer, most likely looking for personal information, passwords, etc.

The above information could not be obtained by an after-the-fact analysis. In particular, the IP address and port of the unknown system could not be determined without a live forensic analysis.

## **Conclusion**

Forensic data gathered from a live system can provide evidence that is not available in a static disk image. Live forensics also operates with different constraints—specifically, the evidence gathered represents a snapshot of a dynamic system that *cannot* be reproduced at a later date. Standards for acceptance are evolving, and legal precedents are still being established. An investigator faces risks that include evidence contamination and facing a court that does not understand the implications of the evidence. Because of the increase in the quantity of digital evidence available in everyday life, it will soon become impossible to acquire *all* disk data relating to a case. The paradigm of live forensics will become the accepted norm.

Live forensic data has been accepted in court cases, and tools exist to gather and analyze this evidence. But the field is still relatively new. Progress in several areas will be essential to increase the usefulness of live forensics, including tools to automate and standardize the process of evidence acquisition and preservation, and presentation tools that allow an investigator to present the facts clearly to a court.

Live forensics needs much better memory analysis tools. Currently, it relies on the—possibly compromised—operating system to provide the list of running processes. Live forensics needs tools to examine the raw memory of a machine and impose a process (and virtual memory) structure on the blocks of memory. These tools are analogous to the static tools that open the raw disk device and impose the file system structure on it to extract files, directories, and metadata.

Most memory analysis tools do little more than extract strings (ASCII for the naïve approach, Unicode for the more sophisticated). None imposes any process structure that provides application-specific information. Unfortunately, limited interest exists in decoding memory images because no one has decoded memory images, and no one has decoded memory images because the interest is limited. Gathering useful time sensitive data and coupling it to fast, real-time analysis will provide new capabilities and insight for digital investigations and incident response.

## Acknowledgements

The author wishes to thank Julie Baker, Rob Joyce, and David Guaspari for their helpful comments.

## References

Frank Adelstein, “The Mobile Forensic Platform,” *Proceedings of the 2002 Digital Forensic Research Workshop (DFRWS 2002)*, Syracuse, NY, August 2002.

Brian Carrier, *File System Forensic Analysis*, Addison-Wesley Professional, March 2005.

Eoghan Casey, *Digital Evidence and Computer Crime*, Academic Press, March 2004.

Dan Farmer and Wietse Venema, *Forensic Discovery*, Addison-Wesley Professional, December 2004.

Vassil Roussev and Golden G. Richard III, “Breaking the Performance Wall: The Case for Distributed Digital Forensics,” *Proceedings of the 2004 Digital Forensics Research Workshop (DFRWS 2004)*, Baltimore, MD, August 2004.

## Biography

Dr. Frank Adelstein is the Technical Director of Computer Security at ATC-NY in Ithaca, NY. He has been working in the area of digital forensics for the past five years. He is the designer of the OnLine Digital Forensic Suite™ tool. Dr. Adelstein received a Gold GCFA certification (GIAC Certified Forensics Analyst) in 2004. Adelstein received a BS Eng. in Computer Engineering from the University of Michigan, and his MS and PhD in Computer Science from The Ohio State University. His post-doctoral work was at Cornell University for the Xerox Design Research Institute. His areas of

interest include networks, security, and distributed systems. He is currently the vice-chair of the Digital Forensics Research Workshop.

© ACM, 2006. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in *Communications of the ACM*, {Vol. 49, No. 2, February 2006} <http://doi.acm.org/10.1145/1113034.1113070>.