

Forensic Analysis

Wietse Venema

wietse@porcupine.org

IBM T.J. Watson Research Center, Hawthorne, NY, USA

Abstract

In this paper I will present lessons learned about persistence of information in file systems and in main memory of modern computers. This is work in progress; the work on main memory was not presented before. The results are based on measurements of a variety of UNIX and Linux systems, but are expected to be valid for other modern operating systems as well.

The treachery of images

About 70 years ago, Rene Magritte made a series of paintings that dealt with the treachery of images. One of those paintings shows an image of a pipe. Below the pipe is a text that says "ceci n'est pas une pipe". This is not a pipe - it's an image of a pipe.

Computer systems are subject to the treachery of images as well. The image on your computer screen is not a computer file - it's only an image on a computer screen. Images of files, processes and network connections are very distant cousins of the actual bits in memory, in network packets, or on disks. The images that you see are produced by layer upon layer of hardware and software. When an intruder controls a machine, any of those layers could be tampered with. Application software can lie, operating system kernels can lie, boot PROMs can lie, and even the software in hard disk drives can lie.

Volatility

Computer information is volatile. The time scale ranges from nanoseconds for processor registers to years for backups and printed information. By merely accessing a file on a computer system one changes the content of main memory, the file's access time stamps, and other information. These side effects can complicate an investigation. On the positive side, these side effects can also reveal past intruder activity that would otherwise have remained unnoticed.

Computer systems tend to spend their life in more-or-less fixed routine behavior. Our measurements show that on specific servers the majority of the files is rarely accessed at all, and that only a limited number of files is modified each day. Some information erodes away quickly due to this routine system behavior. However, our measurements show that deleted file information can persist for days or even weeks.

A trail of deleted information

Computers delete information more frequently than many people realize. Sometimes information is deleted on request by a user. More often, information is deleted implicitly when an application discards some temporary file for its own internal use. Examples are text editor temporary files,

and files in web browser caches. As you use a computer system you leave behind a trail of deleted information.

Information is also deleted as a side effect of system activity that happens in the background. Examples are temporary files in mail system queues or in printer queues. Logfiles are another example of file creation and deletion activity that happens in the background.

To delete or not to delete, that is the question

There is a major difference between deleting information and destroying its contents. With many computer systems, deleted file information remains intact on the disk in unallocated data blocks and in unallocated file attribute blocks, until it is overwritten in the course of other activity.

In 1996, Peter Gutmann presented an excellent paper on the problem of data destruction, with a follow-up in 2001. Peter's concern is with the security of sensitive information such as cryptographic keys and unencrypted data. The best encryption in the world is no good when keys or unencrypted content can be recovered by searching the disk drive or machine memory.

Semiconductor memory chips can be read even after a machine is turned off, but you have very little time. Data on a magnetic disk can be recovered even after it is overwritten multiple times, although the chances of success are becoming vanishingly small as modern disk drives pack the bits more and more closely together. However, as we discuss in this paper, large amounts of information can be recovered without having to use exotic techniques.

Persistence of deleted file attributes

In a first study we looked at the time stamp attributes of deleted files. On UNIX/Linux systems, these file attributes reveal not only when a file was deleted, they also reveal when the file content was last accessed before it was deleted. Some systems also preserve the time of last file content modification. Time stamp attributes, together with other attributes such as file ownership, can reveal a lot about past user activity.

The first measurement was done on a dedicated mail/web/ftp server that runs in a more or less fixed daily routine. The results show a nice gradual decay of the number of deleted file time stamp attributes versus the time since deletion. Typically, half the number of deleted file attributes was overwritten after 20 days. However, there were some fluctuations, and we could still find time stamp attributes of files that were deleted hundreds of days ago.

The second measurement was done on a personal workstation. Here, system usage patterns were dominated by less predictable user activity. This resulted in relatively large fluctuations in the number of deleted file time stamp attributes versus the time since deletion. Again, we found time stamp attributes of files that were deleted hundreds of days ago.

Persistence of deleted file content

When a file is deleted, the content does not accurately reveal when deletion happened. In order to find out how long deleted file content survives before it is overwritten we ran an experiment on a half-dozen machines. Every night an automated job recorded a digital hash of every disk block, as well as the disk block's status: allocated, unallocated, or overhead such as file attribute block. This measurement ran for about five months and produced 100 MBytes of raw data.

The results were quite illuminating. Depending on the type of system we found that deleted file content was overwritten more or less gradually. Half the deleted file content was overwritten after two to five weeks. The number depended the amount of free disk space and on how busy

the machine was. It was in the same order of magnitude as the numbers we found for the deleted file (time stamp) attributes in the previous experiment.

Main memory, first results

In the spirit of the work on deleted file information, we now turn to the question of what happens with information in main memory when the program that uses the information terminates.

All programs that execute on a computer, and all information that is processed by a computer, eventually end up in main memory: program files, data files, temporary files, and of course the operating system itself. Information from running programs may also be found in swap files when the system needs more memory than is physically available.

Although most computer systems lose the content of main memory soon after the machine is turned off, desktop systems are typically turned on for an entire working day, and there is a significant number of computer systems that are turned on permanently.

Long-term memory persistence: file system cache

One important use of main memory is the file system cache (buffer). Whenever a program reads, writes or executes a file, the operating system attempts to keep the contents of that file in main memory, even after the program terminates. File system caching improves system performance by eliminating disk read delays, and is one of the main reasons why computer systems become faster as more memory is added.

Measurements of several UNIX/Linux systems show that frequently accessed files remain in main memory almost permanently. Even information that is accessed only once a day can remain in main memory for many hours. Thus, memory dumps can reveal information about files that were active in the recent and not so recent past.

Short-term memory persistence: private data, deleted files

The other important use of main memory is the storage of private data of running programs. Private memory contains portions of documents, images, input/output buffers, and so on. Private memory is very different from the memory that is used for the file system cache. As long as a program runs, that program has sole access to its private memory. However, once a program terminates, its private memory becomes a prime candidate for reuse.

Measurements on several lightly loaded servers show that private memory is reused relatively quickly after a process terminates. On two lightly loaded servers we found that half of 1Mbyte of private memory was overwritten 200 seconds after program termination. The half-life was only 100 seconds on a personal workstation. Thus, memory dumps reveal very little of the private memory from terminated programs.

Modern operating systems have become quite smart about what memory to preserve (namely, cached file system data) and what memory to reuse first (namely, private data from terminated programs). As was to be expected, deleting a file makes its cached file system information a prime candidate for reuse, just like the private memory from terminated programs. Thus, memory dumps reveal very little of the content from deleted files; such information is more likely to be found on the disk itself.

Summary

In the course of normal and abnormal operation, computer systems generate and delete information frequently. However, this does not mean that information is destroyed immediately.

On specific systems we find that deleted file content and attributes can survive for weeks to months before they are eventually overwritten.

Copies of recently accessed or executed files may stay cached in main memory from hours to days after a program terminates. A system memory dump can therefore provide a legitimate investigator with additional insight into a system's recent history. Of course, this also provides those with less noble intentions with another way to spy on innocent users of computer systems.

References

- Some information about Rene Magritte's work can be found on-line at <http://www.magritte.com/>
- Peter Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory", Sixth USENIX Security Symposium Proceedings, San Jose, California, July 22-25, 1996. http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html
- Peter Gutmann, "Data Remanence in Semiconductor Devices", 10th USENIX Security Symposium, Washington, D.C., August 13-17, 2001. <http://www.cryptoapps.com/~peter/usenix01.pdf>
- Images of magnetic disk patterns can be found in the nanotheatre web pages at <http://www.veeco.com/>.
- Dan Farmer, Wietse Venema, "Coroner's toolkit", available from its author's web sites at <http://www.porcupine.org/forensics/> and <http://www.fish.com/forensics/>
- Dan Farmer, Wietse Venema, Column in Dr Dobb's Journal, <http://www.porcupine.org/forensics/column.html>
- Brian Carrier, "TASK toolkit", available from <http://www.atstake.com/research/tools/>