By SIMSON L. GARFINKEL

AFF: A New Format for STORING HARD DRIVE IMAGES

Most forensic practitioners work with just one or a few disks at a time. A wife might bring in her husband's laptop for imaging a few days before she files for divorce. Police might raid a drug dealer's apartment and seize a PC that's used for contacting suppliers. In these cases it is common practice to copy the drive's contents sector-for-sector into a single file, referred to here as raw copy. Some practitioners bypass the file entirely, and just make a raw copy to a second disk drive that is the same size (or larger) than the original.

Raw images are widely used because they work with practically every disk forensics tool available today. But raw images are not compressed, and can be quite large, even if the drive itself contains very little data.

The obvious way to solve the data storage problem is with a file compressor such as gzip or bzip2. But neither supports random access within a compressed file. Because a forensic tool requires random access in the same manner that a file system requires random access to a physical disk, disk images compressed with a file compressor must be decompressed before they can be used.

A second problem with raw images is the recording of metadata. Because a raw image is a sector-for-sector copy of a target drive, there is no place to store additional information in the file. As a result, information such as the serial number of the target drive, the name of the investigator who performed the acquisition, and even the date the disk was imaged must be stored elsewhere—for example, in a database. But if metadata is not stored in the image file itself, there is a chance it will become separated from the image file and lost, or even confused with the metadata of another drive.

The file format used by the popular EnCase forensic tool overcomes many of the problems inherent with raw images. EnCase stores a disk image as a series of unique compressed pages. Each page can be individually retrieved and decompressed in the computer's memory as needed, allowing random access to the contents of the image file. The EnCase format also has the ability to store metadata such as a case number and an investigator. Unfortunately, the EnCase format is proprietary; although a few vendors have tried to reverse engineer the format to provide for some compatibility, such support is necessarily incomplete.

Faced with this situation, we designed a new file format for our forensic work. Called the Advanced Forensics Format (AFF), this format is both open and extensible. Like the EnCase format, AFF stores the imaged disk as a series of pages or segments, allowing the image to be compressed for significant savings. Unlike EnCase, AFF allows metadata to be stored either inside the image file or in a separate, companion file. Although AFF was specifically designed for use in projects involving hundreds or thousands of disk images, it works equally well for practitioners who work with just one or two images. And in the event the disk image is corrupted, AFF internal consistency checks are designed to allow the recovery of as much image data as possible.

The AFF format is unencumbered by any patents or trade secrets, and the open source implementation is distributed under a license that allows the code to be freely integrated into either open source or propriety programs. We hope that AFF will be adopted by other tool vendors and become a standard format for storing disk images.

An Open, Extensible Format

In order to create a format that will allow for both forward and backward compatibility over an extended time, AFF is partitioned into two layers. AFF's lower *data storage layer* describes how a series of name/value pairs are stored in one or more disk files in a manner that is both operating system and byte-order independent. AFF's upper *disk representation layer* defines a series of name/value pairs used for storing disk images and associated metadata.

The original plan for AFF's lower layer was to use an open source b-tree implementation such as Berkeley DB (BDB), the GNU Database Manager (GDBM), or a similar system. But BDB and GDBM are distributed under the GNU Public License, which is unacceptable to most commercial software developers. (A licensed version of BDB with fewer restrictions can be obtained from SleepyCat software, but not for free.) The Apache SDBM database has no such restrictions, but it creates sparse files that cannot be efficiently copied between machines. Another concern was the layout of information in a b-tree file might be too complex to explain in court, should the need arise.

Instead, we designed a new, simple system based on a repeatable, variable-length structure called an "AFF segment." Each AFF segment consists of a header, a variable-length segment name, a 32-bit flag, a variable-length data payload area, and a segment footer. The segment's length is stored in both the header and footer, allowing for rapid seeking through a file (only the headers and the footers must be read.) The AFF file begins with a file header and ends with a directory that lists all of the segments in the file and their byte offsets from the file's start. If no directory is present, one can be readily constructed by scanning the file from beginning to end, a process that typically takes just a few seconds.

AFF's disk representation layer defines specific segment names used for representing disk information and metadata. For example, "device_sn" is the name of the segment used to hold the disk's serial number, while "date_acquired" holds the time the disk image was acquired. Two special segments are "accession_gid," which holds a 128-bit globally unique identifier that is different each time the disk imaging program is run, and "badflag," which holds a 512B block of data that identifies blocks that cannot be read in the disk image. The metadata can be stored in the same AFF file as the image or in a separate file. Indeed, the schema could even be stored in an XML file.

Tagging bad blocks with a specific pattern is superior to the more common technique of filling bad blocks with ASCII NUL characters because it allows sectors that are unreadable to be distinguished from those that have been manually cleared. On the other hand, we thought the complexity of having a separate map of bad blocks was not warranted.

The image data itself is split into one or more data segments. All data segments must be the same size; this size is determined when the image file is created. Segments are given sequential names—for example *seg0, seg1, seg2*—and continuing to *segn*, where *n* is as large as is necessary to store the disk image. The segment size is stored in a segment called "segsize."

AFF data segments can be compressed with the open source zlib or left uncompressed; whether or not

a data segment is compressed or is encoded in the data segment's 32-bit flag. Compressed AFF files consume less space, but are slower to create and slower to access. The decision to compress or not to compress can be made at runtime, and uncompressed AFF files can easily be compressed or vice versa.

AFFLIB and AFF Tools

AFFLIB is the open source library that implements the underlying AFF system. Rather than forcing the programmer to understand segments, data segments, compression and so on, AFFLIB implements a simple abstraction that makes the AFF image file appear as a persistent name/value database and as a standard file that can be opened, read, and seeked with the *af_open(), af_read()* and *af_seek()* library calls. (In fact, the AFFLIB also supports an *af_write()* call to make it easier to write disk-imaging programs.) If *af_open()* is used to open a non-AFF file, the library defaults to a pass-through mode of operation, allowing AFF-aware programs to work equally well with raw files.

The AFF source code comes with a set of tools including a disk-imaging program (**aimage**), a program for converting AFF metadata into XML (**afxml**), a program for converting raw images to AFF images and back (**afconvert**).

We have also modified Brian Carrier's open source Sleuth Kit to work with AFF-formatted files. The modifications are relatively minor and are likely to be incorporated into the Sleuth Kit's public release. When run interactively, no performance degradation is noticeable—not even on compressed AFF files.

We have been able to store more than one terabyte of disk images in less than 200GB using AFF. We are now working to improve the AFF Tools and the performance of AFFLIB. More information about AFF, including the source code, can be found at www.afflib.org. C

THE AFF FORMAT IS UNENCUMBERED BY ANY PATENTS OR TRADE SECRETS, AND THE OPEN SOURCE IMPLEMENTATION IS DISTRIBUTED UNDER A LICENSE THAT ALLOWS THE CODE TO BE FREELY INTEGRATED INTO EITHER OPEN SOURCE OR PROPRIETY PROGRAMS. WE HOPE THAT AFF WILL BE ADOPTED BY OTHER TOOL VENDORS AND BECOME A STANDARD FORMAT FOR STORING DISK IMAGES.

Simson L. Garfinkel (simsong@acm.org) is a fellow at the Center for Research on Computation and Society at Harvard University, Cambridge, MA. He is also a founder of Sandstorm Enterprises, a computer security firm that develops advanced computer forensic tools used by businesses and governments to audit their systems.