available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow

*Michael Cohen**, *Simson Garfinkel*, *Bradley Schatz*

Australian Federal Police, High Tech Crime Operations, 203 Wharf St., Spring Hill, Brisbane 4001, Australia

Keywords:
Digital forensics
Image
Hard disk Imaging
Digital Evidence Management
Distributed Storage
Distributed Forensic Analysis
Forensic File Format
Evidence Archiving
Cryptography
Forensic Integrity

ABSTRACT

Forensic analysis requires the acquisition and management of many different types of evidence, including individual disk drives, RAID sets, network packets, memory images, and extracted files. Often the same evidence is reviewed by several different tools or examiners in different locations. We propose a backwards-compatible redesign of the Advanced Forensic Format—an open, extensible file format for storing and sharing of evidence, arbitrary case related information and analysis results among different tools. The new specification, termed AFF4, is designed to be simple to implement, built upon the well supported ZIP file format specification. Furthermore, the AFF4 implementation has downward comparability with existing AFF files.

1. Introduction

Storing and managing digital evidence is becoming increasingly more difficult, as the volume and size of digital evidence increases. Evidence sources have also evolved to include data other than disk images, such as memory images, network images and regular files. Preserving such digital evidence is an important part of most digital investigations (Carrier and Spafford, 2004), and managing the evidence in a distributed organization is now emerging as a critical requirement.

This paper presents a framework for managing and storing digital evidence. We first examine existing evidence management file formats and outline their strengths and limitations. We then explain how the proposed Advanced Forensics Format (AFF4) framework extends these efforts into a universal evidence management system. The detailed description of the AFF4 proposal is then followed by concrete real world use cases.

1.1. Prior work

In recent years there has been a steady and growing interest in the actual file formats and containers used to store digital evidence. Early practitioners created exact bit-for-bit copies (commonly referred to as "dd images"). More recently, proprietary software systems for making and authenticating "images" of digital evidence have become common (e.g. B.S. NTI Forensics Source, 2008; Ilook investigator, 2008; Guidance Software, Inc., 2007). PyFlag (Cohen, 2008a) introduced a "seekable gzip" format that allowed disk images to be stored in a form that was compressed but allowed random access to evidence data necessary for forensic analysis.

* *Corresponding author*. Tel.: +61 732221361.
E-mail address: scudette@gmail.com (M. Cohen).

1742-2876/$ – see front matter
doi:10.1016/j.diin.2009.06.010

The Expert Witness Forensic (EWF) file format was originally developed for Encase (Guidance Software, Inc., 2007), but then adopted by other vendors (Kloet et al., 2008). The EWF file format similarly compresses the image into 32 kb chunks which are stored back to back in groupings inside the file. The format employs tables of relative indexes to the compressed chunks to improve random access efficiency. EWF volumes have a maximum size limit of 2 Gb and therefore usually split an image across many files. EWF provides for a small number of predefined metadata fields to be stored within the file format.

The Advanced Forensic Format (AFF) expanded on this idea with a forensic file format that allowed both data and arbitrary metadata to be stored in a single digital archive (Garfinkel et al., 2006).

Both the AFF and EWF file formats are designed to store a single image, and any metadata that implicitly refers to that image such as sector size and acquisition date. Unlike EWF, AFF employed a system to store arbitrary name/value pairs for metadata, using the same system for both user-specified metadata and for system metadata, such as sector size and device serial number. For example, `Aimage`, the AFF hard disk acquisition tool, not only stores the image, but additionally stores a description of the tool itself, the version of AFFLIB used to create the image, the computer on which the image was made, the operator of the tool, the user supplied parameters supplied to the tool.

Schatz proposed a Sealed Digital Evidence Bags architecture, facilitating composition of evidence and arbitrary evidence related information, through a simple data model and globally unique referencing scheme (Schatz and Clark, 2006).

1.2. *This paper*

An important advance of this work is the introduction of storage transformation functions to the forensic storage container. Prior works simply focused on forensically sound storage of bit-streams, leaving the necessary activities of translating low level storage into higher level abstractions at the aggregate block (i.e. RAID), volume, and filesystem layers in the domain of analysis tools, as transiently constructed artifacts. In contrast AFF4 has mechanisms for describing transformation in a flexible and concise way, allowing users to view multiple transformations of the same data with little additional storage cost. This mechanism is an important enabler for inter-operable forensic tools. For example, carved files may be described in terms of their block allocation sequences from an image, rather than requiring the carved file to be copied again.

This paper extends previous work on the Advanced Forensic Format (AFF) by taking many of the concepts developed and designing a new specification and toolset. The AFF4 format is a complete redesign of the architecture. The new architecture is capable of storing multiple heterogeneous data types that might arise in a modern digital investigation, including data from multiple data storage devices, new data types (including network packets and memory images), extracted logical evidence, and forensic workflow. The AFF4 format extends the format to make it the basis of a global distributed evidence management system.

We call the new system AFF4, and use the phrase AFF1 to refer to the legacy system developed by Garfinkel et al.[1] The publicly released AFF4 implementation, is able to read existing AFF files.

2. The need for an improved forensic format

AFF1's flexibility came from a data model of forensic data and metadata stored as arbitrary name/value pairs called *segments*. For example, the first 16 MB of a disk image is stored in a segment called page0, the second 16 MB in a segment called page1, *etc.* Because of this flexibility, it was relatively easy to extend AFF1 to support encryption, digital signatures, and the storage of new kinds of metadata such as chain-of-custody information (Garfinkel, 2009).

2.1. *AFF limitations*

We observed a number of practical problems in the underlying AFF1 standard and Garfinkel's AFFLIB implementation:

- While AFF1's design stores a single disk image in each evidence file, modern digital investigations typically involve many seized computers or pieces of media.
- The data model of AFF1 enabled storing metadata related to the contained image as (property, value) pairs. This data model does not, however, support expressing arbitrary information about more than one entity.
- AFF1 has no provision for storing memory images or intercepted network packets.
- AFF1 has no provisions for storing extracted files that is analogous to the EnCase "Logical Evidence File" (L01) format, or for linking evidence to web pages.
- AFF1's encryption system leaks information about the contents of an evidence file because segment names are not encrypted.
- AFF1's default compression page size of 16 MB can impose significant overhead when accessing NTFS Master File Tables (MFT), as these structures tend to be highly fragmented on systems that have seen significant use.
- Although the AFF1 specification calls for a "table of contents" similar to the Zip (Katz, 2007) "central directory" that is stored at the end of AFF files, Garfinkel never implemented this directory in the publicly released AFF1 implementation, AFFLIB. As a result, every header of every segment in an AFF file needs to be read when a file is opened. In practice this can take up to 10–30 s the first time a large AFF file is opened.
- AFF1's bit-level specification is essentially a simple container file specification. Given that there are other container file specifications that are much more widely supported with both developer and end-user tools, it seemed reasonable to migrate AFF from its home-grown format to one of the existing standards.

[1] Although Garfinkel never changed the AFF bit-level specification, Garfinkel released AFFLIB implementations with major version numbers 1, 2 and 3. We therefore call our system AFF4 to avoid confusion.