Functional Access to Forensic Disk Images in a Web Service

Kam Woods
UNC Chapel Hill
CB #3360, 100 Manning Hall
Chapel Hill, NC 27599-3360
919-962-8366
kamwoods@email.unc.edu

Christopher A. Lee
UNC Chapel Hill
CB #3360, 100 Manning Hall
Chapel Hill, NC 27599-3360
919-962-8366
callee@ils.unc.edu

Oleg Stobbe
University of Freiburg
Hermann-Herder Str. 10
79104 Freiburg, Germany
oleg.stobbe@rz.uni-freiburg.de

Thomas Liebetraut
University of Freiburg
Hermann-Herder Str. 10
79104 Freiburg, Germany
thomas.liebetraut@rz.uni-freiburg.de

Klaus Rechert
University of Freiburg
Hermann-Herder Str. 10
79104 Freiburg, Germany
klaus.rechert@rz.uni-freiburg.de

ABSTRACT

We describe a hybrid approach for access to digital objects contained within forensic disk images extracted from physical media. This approach includes the use of emulation-as-a-service (EaaS) to provide web-accessible virtual environments for materials that may not render or execute accurately on modern hardware and software, and the use of digital forensics software libraries to produce web-accessible file system views to support single-file access and provide visualizations of the file system.

General Terms

Frameworks for digital preservation; preservation strategies and workflows

Keywords

Emulation, access, digital forensics

1. INTRODUCTION

Support for meaningful use of digital objects often requires retention of the environment (or aspects of the environment) in which they were produced. This can help to reproduce significant properties of the digital objects [2], as well as reflecting essential contextual information.

For materials acquired on fixed and removable digital media, addressing this need begins with acquiring a complete disk image, which is a block-by-block copy of the disk's storage. No prior knowledge of the operation system (OS) or file system on the disk is required to perform the acquisition. Similarly, one can search for patterns within the bitstream (e.g. email addresses, credit card numbers, phone numbers) without necessarily knowing or having software support for the original OS or file system [1].

Analysis and description tasks can require mounting of the original file system. These include navigation of the files and folders; extraction of specific files or folders; extraction of file system metadata; and reporting the number and types of files on disk. Additional digital curation actions also require software that can recognize, access and render information from specific file formats. These include file characterization, validation, metadata extraction and visual inspection.

File systems and file formats are subject to obsolescence, and digital curation professionals often process born-digital materials that are not supported by contemporary computing environments. One response to this challenge is to install dedicated software (applications or complete operating systems) on the machine being used to process the materials, or to consolidate these tools into a specialized environment. An example of this is the BitCurator environment[1], a suite of open source digital forensics and data analysis tools to help collecting institutions (libraries, archives, and museums) process born-digital materials. This environment, developed through a series of grants from the Andrew W. Mellon Foundation, has been customized to work with many obsolete file systems and file types. It also contains software for the creation of forensic disk images; analysis of files and file systems; extraction of file system metadata; identification of sensitive information; and identification and removal of duplicate files.

There is always the possibility of acquiring disks with file systems and files that are not supported by the available tools. One also cannot assume that end users will be running specialized tools on their local machines. An alternative access strategy is emulation: enabling the user to boot and

[1]BitCurator, http://www.bitcurator.net/

interact with an original operating system, or attaching the disk as a secondary drive to an emulated environment typical of the era in which it was produced. Emulation-as-a-Service (EaaS) simplifies this process for end users by providing access to pre-configured emulation environments within a web browser.

We present an approach to accessing operating systems and file systems contained in disk images using both EaaS[2] and a dedicated web application to generate views into non-live file systems. For public (or semi-moderated) access, redaction of sensitive content is often required. We describe a traceable redaction workflow and implementation for restricted functional access to disk images, supporting different access levels depending on the requester's role.

2. RELATED WORK

Capture and analysis of disk images from fixed and removable media is a mainstay of digital forensics practice. The need to quickly analyze large quantities of digital information has led to the development of several modular open-source tools and platforms to parse file system contents and identify and analyze features of interest within the file systems [1].

The development of open-source digital forensics tools to manipulate common disk image file formats (along with tools to create and export from them) increases the attractiveness of digital forensics tools to collecting institutions. These include *libewf*, an open source library to create and manipulate files in the widely-used Expert Witness Format [4].

3. ACCESS WORKFLOW DESCRIPTION

For purposes of this discussion, we assume that one has already created a disk image along with a description of the disk's technical environment (e.g. source descriptions, size in bytes, file system(s) present). This information is used to make decisions about the access environment and enable preparation of any surrogate – for example, if there is data within the disk image that requires redaction.

3.1 Preparation

It is important to distinguish between two distinct access modes: interacting with the disk image as a bootable system disk (e.g. if the disk contains an operating system); and attaching the disk as a secondary disk to an emulated environment. In the latter case, no further measures are required. The former case requires a hardware generalization process that we describe below.

First, a description of the original hardware environment associated with the disk image is examined to identify the correct emulator configuration (or locate a similar emulated system prepared previously). Emulators typically provide only a limited selection of hardware system components – usually popular devices with broad driver support. For systems from the 1980s and 1990s, for example, common ISA-bus hardware devices with virtualization support include the Soundblaster 16 and AdLib sound cards, the NE2000 network adapter, and the Cirrus VGA graphics adapter.

[2]bwFLA EaaS, `http://bw-fla.uni-freiburg.de`

To recreate a system associated with specific hardware, additional hardware drivers may need to be installed or replaced – at least if full functionality is required. This process may require certain changes to the disk image. These changes, however, must not be applied directly to the (forensic format) disk image, but have to be kept as a separate change-set, which supports tracking of (technical) modifications both on a block and file system level.

The result of the preparation process is a set of technical changes required to run on a generic emulated computer system. While the acquired image may be altered, this generalization process also comes with benefits: the machine setup is fully documented and understood, and hardware dependencies are explicit and can serve as a preservation and planning guide for emulating other systems in the future.

3.2 Redaction & Dissemination

Bootable system disks are more likely to contain items that require redaction, including personally identifying and sensitive information within documents explicitly produced by the original user(s), and other data retained via the normal operation of the operating system and file system.

As an example, Windows-based systems retain information corresponding to various user activities, including passwords (which may not be well encrypted in earlier versions of the OS), lists of recently viewed documents, devices that were attached to the original system, and - potentially - sensitive data including online credentials and encryption keys. Depending on the version of the OS used when the system was active, this information may appear in the Registry, in the hibernation file used for fast resume on system wakeup, and in unallocated areas of the disk.

In past publications, we have shown how open-source digital forensics tools such as Simson Garfinkel's *bulk_extractor* may be incorporated into archival workflows, allowing users preparing collections for access to redact both at the block level on disk, and to restrict access to individual files [3].

Rather than storing raw disk images, many collecting institutions are using forensic disk image formats, such as EWF, which can compress the data, as well as embedding integrity checks and various forms of metadata. EWF files cannot be redacted in-place without compromising consistency checks internal to the file format. One workaround is to export the raw data from the EWF image, redact the relevant blocks (or mount and redact specific files or directories), and create a new EWF file using the redacted raw image. There are cases when this approach may not be desirable, because it complicates the provenance record of the stored data.

Alternatively, features identified by *bulk_extractor* may be linked to individual file items and recorded in an annotated Digital Forensics XML (DFXML) file. Working with *libewf*, an open-source software library, it is possible to create a synthetic listing of the contents of the file system within an EWF file that elides any file or directory item within the DFXML file that is marked as containing restricted material.

For non-emulated access on the Web – viewing the contents

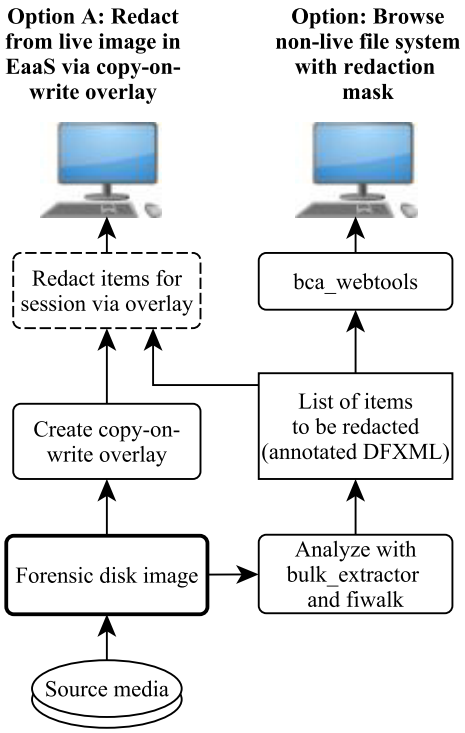

Figure 1: Simplified workflow showing redaction options for emulation and browsing access.

of the file system in a simulated directory structure within a web page – this file may be consulted to determine whether a given file or directory should be presented to the user as a link. This effectively "blacklists" files containing restricted, sensitive, or private information from public access.

Creation of a "surrogate" EWF image using an exported, redacted raw image from the original EWF source is an obvious approach for facilitating restricted access. However, the storage and time requirements associated with creating altered copies of original disk images may be prohibitive – storage alone will effectively double unless the original image is discarded.

As an alternative, the blacklist annotations to the DFXML representation of the file system may be passed to the emulation tool to modify the file system immediately prior to user access, deleting file items and scrubbing unallocated areas prior to enabling user access. Some access options when providing redaction services for forensic disk images are shown in Figure 1.

3.3 Emulation-based Access

Web-based access to born-digital archival materials is often restricted to individual files that have been specifically selected for access. These files may be normalized (e.g. converting Microsoft Word to PDF/A), with the only context for the *original* environment being in the archival metadata that accompanies the file.

This can degrade the access experience in several ways. Executable content may not run on modern systems, or may depend on hardware that is not accommodated by (or simulated by) modern device drivers. Second, the user may be more interested in the original structure and organization of the content than the content itself. Finally, there may be features or limitations of the production environment (the bootable operating system) that are of interest with respect to their influence on the documents or media produced.

Emulated environments can provide a view of the original production environment, but have traditionally faced various hurdles, including lack of computing power on the end-user's system and lack of expertise in installing and configuring required software. Emulation-as-a-Service addresses these limitations by offloading the computational requirements to a hosted service and providing users with "one-click" access to bootable environments within a web browser. Emulation platforms such as QEMU provide access to a range of operating system environments and disk image formats, but support for formats most common in forensic disk imaging did not previously exist. In the following section, we describe a mechanism (including a novel QEMU block driver) to enable access to forensic disk images in EWF format.

3.4 Implementation

A disk image captured as an EWF file is effectively read-only. Any deliberate alterations to the content of the image will produce error warnings in libraries capable of reading the contents; these changes will cause embedded cyclic redundancy checks to fail.

To use the EWF image in an emulation setup, a writeable disk is required. As a first step we create a writeable *overlay file* that forwards read operations for any unmodified block to the original EWF file. Write operations are captured and only written to the overlay file. This process is known as *copy-on-write*. Subsequent reads of such modified blocks are severed from the overlay file. This mechanism allows data modifications to be stored separately, independent of the original digital object during an emulation session. This allows each digital object to be retained in its preserved, unmodified state. After an emulated session the overlay-file can either be discarded or kept for future use or analysis.

To achieve this we have implemented an EWF QEMU block driver to enable access using QEMU's disk image handling tools and to make use of QEMU's QCOW2 container format [3]. The QCOW2 format allows one to store all changed data blocks and the respective metadata for tracking these changes in a single file. To define where the original blocks (before copy-on-write) can be found, a *backing file* definition is used. QEMU's Block Driver API provides a continuous view on this QCOW2 container, transparently choosing either the backing file or the copy-on-write data structures as source.

As any block format is allowed in the backing file of a QCOW2 container, the backing file can itself be a QCOW2 container.

[3]The QCOW2 Image Format, `https://people.gnome.org/~markmc/qcow-image-format.html`, last access 4/8/15.

This allows "chaining" a series of modifications as copy-on-write files that only contain the actually modified data. One can use this feature to make individual changes to the original environment citable and accessible, for instance, to provide access to a disk's redacted version.

This overlay concept and its implementation does not depend on a specific emulator (such as QEMU). It may be adapted to work with any emulation platform that provides appropriate access. Listing 1 shows an example creating the overlay file `ewf-overlay.cow` using the backing file `ewf-demo.E01`.

Listing 1: Example creating a QCOW2 overlay on top of a EWF file.

```
qemu-img create \
 -f qcow2 \
 -o backing_file=ewf-demo.E01,backing_fmt=ewf \
 ewf-overlay.cow
```

To make use of the overlay file with an arbitrary emulator (including emulators with no native QCOW2 support) the raw payload needs to be exposed. This can be achieved by "fusing" the QCOW2 container to expose its raw content as a synthetic continuous file. Read and write operations are intercepted by the FUSE[4] file system layer and translated to appropriate QCOW2 read/write operations.

Listing 2 uses `qemu-fuse` to expose the raw disk image. The resulting file `raw-content/ewf-overlay.cow` contains the bit-exact copy of the original physical disk without any additional metadata added by the EWF format or QCOW2 container and therefore can be attached directly to an emulator.

Listing 2: Expose raw disk content using qemu-fuse

```
qemu-fuse ewf-overlay.cow raw-content/
```

Capturing changes at the lowest possible layer (the block layer) has specific technical advantages compared to higher layers (e.g. file system). First, this approach is independent of the hardware medium (disregarding vendor-specific storage areas on modern devices that have no effect on file system access), and does not depend on any operating system or file system encoded on the device. Second, the required metadata is simple and relatively easy to understand; reconstruction of the file is possible even without access to the original tools. Listing 3 shows metadata of an unmodified overlay file, with all blocks mapped to the (original) backing file.

Listing 3: The block mapping table before modification of the overlay file

```
Offset  Length      Mapped to   File
0       0x1f400000  0           ewf-demo.E01
```

Listing 4 shows metadata after modification[5]. Several blocks have been changed on the disk and are now mapped to the overlay file.

Listing 4: An excerpt from the block mapping table after modification of the overlay file

```
Offset    Length      Mapping   File
0         0x10000     0x60000   ewf-overlay.cow
0x10000   0x10000     0x10000   ewf-demo.E01
0x20000   0x10000     0x70000   ewf-overlay.cow
0x30000   0x10000     0x30000   ewf-demo.E01
0x40000   0x10000     0x50000   ewf-overlay.cow
0x50000   0x620000    0x50000   ewf-demo.E01
0x670000  0x10000     0x80000   ewf-overlay.cow
0x680000  0x1ed80000  0x680000  ewf-demo.E01
```

While critical to the implementation, these details are not visible to the end user. The user sees only an environment that can be navigated, modified, and otherwise interacted with, while the underlying disk image (the preservation object) remains unchanged.

4. USE CASES & EVALUATION

As outlined in the previous sections, we envision two basic use cases: a user browsing the file system of a forensic disk image via a web-interface, and a user interacting with a booted file system or secondary storage device via an emulated environment rendered within a web browser.

Both approaches support interaction with forensic disk images by providing access to the underlying file system(s) using existing open source libraries to read the contents of the disk image format.

4.1 Using an EWF Image as Boot Disk

To evaluate the capabilities of our tools and workflow we have chosen a real use case, demonstrating the image preparation process, i.e. a technical generalization to be used with an appropriate emulator.

The Vilem Flusser Archive owns a personal computer associated with the production of a software titled "Flusser-Hypertext". This computer contains a rare working copy of the software which is dependent on the obsolete authoring system HyperCard. The disk image has been acquired[6] from an Apple Mac Performa 630 containing a 270 MB IDE disk. The goal was to enable web-based access to the Flusser-Hypertext through the archive's web site.

Using the acquired disk image directly with an emulator failed. The original machine used a hardware-related extension (A/ROSE) that is not supported by the emulator used and prevented the system to start properly. A simple solution was to boot the system with all extensions disabled and to delete the A/ROSE extension file from the system's extensions folder. The result of this process is an overlay-file that is bootable and useable with an emulator. The overlay's file size is 823 KB and contains 7 changed blocks (block size was set to 1024 bytes). However, simply booting the (unmodified) file system results in 3 changed blocks.

4.2 Redaction for Public Access

The disk image is now fully functional in an emulation scenario. However, it is not yet suitable for public access. As sensitive private data was found on the disk image, these

[4]FUSE: Filesystem in Userspace, `http://fuse.sourceforge.net/`

[5]In this case a MS-DOS 6.2 system has been booted and a directory has been created on the disk

[6]The original acquisition was performed using `dd` without forensic tool support. We have reacquired the raw disk image as an EWF image.

files have to be removed, and a second overlay file has to be produced.

In case of the Flusser Mac the archive provided a list of files that are not suitable for public access. These files have been removed from the file system on a second overlay file, which is now accessible through the archive's web site[7] and citable. In general, the redacted version of the disk image is inextricably linked to the original image, such that any action of the redaction process can be audited.

4.3 Access using EaaS

Once an overlay file for public access has been created, it can be published using the EaaS framework. In a typical EaaS setup, the emulator runs either on a local computing cluster or using a cloud computing service (such as Google Cloud). Disk image storage, description, and publication is managed by the respective preservation institution.

To securely publish a redacted disk image, only a standard web server[8] is required. Ideally, the redacted overlay-file points to a local backing file that is not accessible through the web server. Properly implemented, this ensures that a user visiting the website cannot exploit a vulnerability of the server-side software to read data directly from the overlay (for example, to examine blocks from the original disk image that have been scrubbed, or files that have been "deleted" from the image via the overlay).

The EaaS service requires a `binding` configuration as part of the technical metadata, defining the data source' to be configured as an emulated machine's `drive`. Listing 5 shows an example of an EaaS configuration. If no redaction is required, or the emulator access is not public, a pointer to the EWF file (e.g. an HTTP link) in the bindings section is sufficient.

Listing 5: Metadata defining data resources and emulator medium mapping

```
[...]
 <drive>
  <data>binding://main_hdd</data>
  <iface>ide</iface>
  <bus>0</bus>
  <unit>0</unit>
  <type>disk</type>
  <boot>true</boot>
  <plugged>true</plugged>
 </drive>

 <binding id="main_hdd">
  <url>https://.../diskImage.pub</url>
  <access>cow</access>
 </binding>
[...]
```

In both cases these images can be cited (e.g. using HDL) and functionally accessed [9].

[7] `http://www.flusser-archive.org/`

[8] HTTP range request support is required to avoid transferring the complete disk image to the emulator's site.

[9] Functional access to the Flusser machine. `http://hdl.handle.net/11270/2b87de90-37dc-4d66-a9e6-546a80b0b261`

5. FUTURE WORK

Some of the uncertainty associated with handling disk images extracted from legacy media – particularly when they contain bootable operating systems – is derived from a lack of sufficient description of the technical environment in which they were produced. Providing guidelines for how those technical environments should be described is paramount in supporting the contextualization and generalization process. In future efforts, we intend to provide additional guidance on factors related to both the hardware and operating system that should (at a minimum) be recorded.

Some aspects of this process may be automated, particularly when working with operating systems such as Windows, OS X, and earlier version of the Macintosh operating system that record hardware characteristics in well-documented locations.

We also plan to further explore the relationships between EaaS and navigation of disk image file trees in a web browser [5]. We plan to examine options for creating richer, more unified interfaces to allow users to examine metadata related to disk images, search the contents of images prior to accessing them directly, and browse to EaaS instances from within existing archival access environments.

6. CONCLUSION

We have described a series of methods to provide web-based access to disk images captured in forensic formats; through an emulation system that can be accessed using a modern web browser, and by browsing views of the file system directly within a webpage. These approaches address an important need among collecting institutions: allowing users visiting their website to interact with legacy operating systems and file systems contained in disk images extracted from legacy media, without requiring them to install software or understand the technical details required to recreate a functional environment.

7. REFERENCES

[1] S. L. Garfinkel. Digital media triage with bulk data analysis and bulk extractor. *Computer Security*, 32(C):56–72, Feb. 2013.

[2] M. Hedstrom and C. A. Lee. Significant properties of digital objects: definitions, applications, implications. In *Proceedings of the DLM-Forum 2002*, pages 218–227. Office for Official Publications of the European Communities, 2002.

[3] C. A. Lee and K. Woods. Automated redaction of private and personal data in collections: Toward responsible stewardship of digital heritage. In *Proceedings of The Memory of the World in the Digital Age: Digitization and Preservation*, pages 298–313, New York, NY, USA, 2012. UNESCO.

[4] J. Metz. EWF specification – Expert Witness Compression Format specification. https://github.com/libyal/libewf/wiki, 2006.

[5] S. Misra, C. A. Lee, and K. Woods. A Web Service for File-Level Access to Disk Images. http://journal.code4lib.org/articles/9773, 2014.