2008-06

Automated metadata extraction

Migletz, James J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AUTOMATED METADATA EXTRACTION

by

James Migletz

June 2008

Thesis Advisor:	Simson Garfinkel
Second Reader:	Kevin Squire

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Automated Metadata Extraction			**5. FUNDING NUMBERS**
6. AUTHOR(S) James Migletz			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Metadata is data that describes data. There are many computer forensic uses of metadata and being able to extract metadata automatically provides positive forensic implications. This thesis presents a new technique for batch processing disk images and automatically extracting metadata from files and file contents. The technique is embodied in a program called fiwalk that has a plug-in architecture allowing new metadata extractors to be readily incorporated. Output from fiwalk can be provided in multiple formats such as ARFF and text. The plug-ins created for this thesis include one created by Simson Garfinkel for extracting metadata from .jpeg files, two for Microsoft Office documents (one for prior to Office 2007 release and one for Office 2007 release), and a default plug-in for extracting metadata from .gif, .pdf, and .mp3 files. To better understand the metadata available in common file formats such as .doc, .docx, .odt, .pdf, .mp3, .mp4, .jpeg, .tiff, and .gif, an examination of these formats is provided.

14. SUBJECT TERMS Metadata, Metadata Extraction, Fiwalk, WV, Libextractor, File Formats, ARFF			**15. NUMBER OF PAGES** 83
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AUTOMATED METADATA EXTRACTION

James J. Migletz
Major, United States Marine Corps
B.S., Northwest Missouri State University, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2008

Author: James J. Migletz

Approved by: Simson Garfinkel
Thesis Advisor

Kevin Squire
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Metadata is data that describes data. There are many computer forensic uses of metadata and being able to extract metadata automatically provides positive forensic implications. This thesis presents a new technique for batch processing disk images and automatically extracting metadata from files and file contents. The technique is embodied in a program called fiwalk that has a plug-in architecture allowing new metadata extractors to be readily incorporated. Output from fiwalk can be provided in multiple formats such as ARFF and text. The plug-ins created for this thesis include one created by Simson Garfinkel for extracting metadata from .jpeg files, two for Microsoft Office documents (one for prior to Office 2007 release and one for Office 2007 release), and a default plug-in for extracting metadata from .gif, .pdf, and .mp3 files. To better understand the metadata available in common file formats such as .doc, .docx, .odt, .pdf, .mp3, .mp4, .jpeg, tiff, and .gif, an examination of these formats is provided.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ABBREVIATIONS AND ACRONYMS

AAC	Advanced Audio Coding
AFF	Advanced Forensic Format
ARFF	Attribute Relation File Format
CDI	Customer Data Integration
DGI	Domex-Gateway Interface
DOC	Microsoft Word file extension
DOCX	Microsoft Word 2007 file extension
DVI	Digital Visual Interface
ELF	Executable and Linkable Format
EXIF	Exchangeable Image File
EXT2/EXT3	Second and Third Extended File Systems
FAT	File Allocation Table
FIWALK	File Inode Walk
GDS	Google Desktop Search
GIF	Graphics Interchange Format
GSA	Google Search Appliance
HTML	Hypertext Markup Language
ID3v1	Identify an MP3
MAC	modified, access and creation/change times
MD5	Message Digest algorithm 5
MPEG	Moving Picture Expert Group
NTFS	New Technology File System
ODF	Open Document Format
OLE	Object Linking and Embedding
OOX	Office Open Format
PDF	Portable Document Format
PDM	Product Data Management
PIM	Product Information Management
PNG	Portable Network Graphics
PPT	Microsoft PowerPoint file extension
PPTX	Microsoft PowerPoint 2007 file extension
RID	Relationship Identifier
RIFF	Resource Interchange File Format
RSIDR	Revision Identifier (paragraph and default)
SDK	Software Development Kit
SDT	Structured Document Tag
SHA	Secure Hash Algorithm 1
SQL	Structure Query Language
TIFF	Tagged Image File Format
TSK	The Sleuthkit
UFS	Unix File System
UTF -16	16 bit Unicode Transformation Format

XLS	Microsoft Excel file extension
XLSX	Microsoft Excel 2007 file extension
XML	Extensible Markup Language

GLOSSARY

Attribute Relation File Format (ARFF) – provides a standard way of representing data sets that consist of independent, unordered instances without involving relationships between the instances. The machine learning software `Weka` accepts ARFF files as a standard input format.

Autopsy – a front-end browser for Sleuthkit created to make the digital forensic analysis process easier for the user. (See Sleuthkit).

data mining – the process of identifying patterns in data. The process must be at a minimum semi-automated.

doc – Microsoft Word file extension

docx – Microsoft Word 2007 file extension

docx_extractor – fiwalk plug-in written in Python to extract metadata from Office Open formatted documents. Includes .docx, .xlsx, and .pptx documents.

Domex-Gateway Interface (DGI) – means by which the plug-ins communicate with fiwalk. Fiwalk puts the data into a file in the file system; the plug-in reads the file and returns the data as a series of name:value pairs.

EnCase – a popular forensics tool produced and maintained by Guidance Software.

exif – metadata extraction tool that extracts metadata from .jpeg files. EXIF also refers to the Exchangeable Image File format.

Extensible Markup Language (XML) – a general purpose specification for defining markup languages.

file ascription – associating a file to an owner based on the metadata available from the file.

fiwalk (File Inode Walk) – an application written by Simson Garfinkel that retrieves information from disk partitions found on disk images. The application relies on the Sleuthkit programmer's interface to locate all of the files and orphaned inodes found in a given disk image. Fiwalk's plug-in architecture makes it a very suitable format for utilizing the metadata extraction tools.

Graphics Interchange Format (GIF) – a protocol invented by CompuServe in 1990 that was designed for the on-line transfer and exchange of raster graphic data that is independent of the hardware used.
Google Desktop Search (GDS) – a local searching tool that allows users to index and search the contents of their computers. GDS offers the capability to index and search for files using metadata for file types such as multimedia files that do not generally contain content that can be readily indexed.

Google Search Appliance (GSA) – indexes metadata stored in documents and makes data available for retrieval at search time.

Hypertext Markup Language (HTML) – one of the most popular markup languages for web pages. HTML provides a way to represent the structure of text-based information in a document.

JPEG – one of the most popular file formats used today for the representation of digital photographs.

jpeg_extract – fiwalk plug-in written in Java and C++ to extract metadata from .jpeg using the metadata extraction tool Exif.

libextractor – open source tool that extracts metadata from the following types of formats MP3, Ogg, Real Media, MPEG, RIFF (avi), GIF, JPEG, PNG, TIFF, HTML, PDF, PostScript, Zip, OpenOffice.org, StarOffice, Microsoft Office, tar, DVI, man, Deb, elf, RPM, and asf.

Libextract-plugin – fiwalk plug-in written in Java to extract metadata from .gif, .mp3, .pdf, .png, and .tiff file types using the metadata extraction tool libextractor.

Message Digest algorithm 5 (MD5) – a 128 bit hash function designed by Ron Rivest in 1991.

metadata – data that describes data. File metadata might include creator, creation time, modify time, access time, file modifier, and revision number.

mp3 – a compressed, lossy, perceptual coding scheme format that is part of the Moving Picture Expert Group (MPEG) set of standards for music encoding.

Open Document Format (ODF) – an open, license-free, and clearly documented file format released by OpenOffice.org as an alternative to the document file formats developed by Microsoft. ODF is based on XML, which allows the representation of any complex data type as a recursive tree-structured document consisting of names, attributes, values, and other trees. ODF utilizes a single document represented by multiple XML files bundled together into a single ZIP archive. ODF has been accepted as both OASIS

(Organization for the Advancement of Structured Information Standards) and ISO standards.

odf_extractor – fiwalk plug-in written in Python to extract metadata from Open Office formatted documents. Includes .odt, .ods, and .odp documents.

Office Open Format (OOX) – The Microsoft Office 2007, XML-based document file format. OOX is a ZIP archive file consisting of multiple XML document elements. Microsoft refers to the .docx, .xlsx, and .pptx files as packages, with each file within the archive being referred to as a part.

Portable Document Format (PDF) – file format introduced by Adobe. The primary goal of the PDF is to allow users to easily exchange and view unmodifiable electronic documents.

ppt – Microsoft PowerPoint file extension

pptx – Microsoft PowerPoint 2007 file extension

Revision Identifier for a paragraph (rsidR) – values from Office Open documents used to identify the editing session in which a paragraph was added to a main document. An rsidR value should be unique, and instances with the same value within a single document indicate that the regions were modified during the same editing session.

Revision Identifier default (rsidRDefault) – associated with the rsidR attribute of an Office Open document. An rsidRDefault attribute is used for all runs within a paragraph in which an rsidR is not defined.

Secure Hash Algorithm 1 (SHA 1) – a secure hashing algorithm designed by NIST and NSA for use in digital signatures.

Sleuthkit (TSK) – an open source, Unix-based forensics tool, consisting of over 20 command-line tools used for analyzing disk and file system images for evidence. Sleuthkit is maintained by Brain Carrier.

StoreItem id – identifier used to distinguish between content pieces and to ensure that the correct data is bound.

Structured Document Tag (SDT) – node created when content control is added to a document. The two sections within structured document node are properties and content sections.

Weka – a collection of machine learning algorithms and data preprocessing tools. Accepts ARFF formatted files as one of the input file types.

word_extract – fiwalk plug-in written in Java to extract metadata from pre-2007 Microsoft Office documents using wvSummary. Includes .doc, .xls, and .ppt documents.

wvSummary – wvWare script that prints out the metadata from Microsoft Office documents.

wvWare – command-line driven application from the wvLibrary that contains helper scripts including wvSummary.

xls – Microsoft Excel file extension

xlsx – Microsoft Excel 2007 file extension

zip – compressed file format

ACKNOWLEDGMENT

I would like to thank my wife Diane and my kids Jimmy and Miranda for their unwavering support and confidence while I attended school and worked on this thesis. Without their support and understanding, I would not have accomplished everything I did over the last two years.

I would also like to thank Daniel Huynh for his willingness to work with me on our research and writing efforts and for his assistance in the many classes that we attended together.

Additionally, I owe thanks to Dr. Kevin Squire for his volunteering to act as my second reader. His selflessness helped me in crafting the best thesis I could write.

Finally, I would like to thank Dr. Simson Garfinkel for his patience, guidance and support in helping me shape the scope of this thesis, and for the assistance he provided in the research, writing, and programming efforts. Without his help, completing the thesis would have been significantly more difficult if not impossible.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Currently the intelligence and law enforcement communities rely on trained experts to conduct media exploitation of devices captured either on the battlefield or through investigative work. The analysis can require weeks and in some cases months to complete. For individuals relying on current information to plan and execute their next move, the wait is a severe impediment to their efforts.

Providing the intelligence and law enforcement personnel with an automated means of exploiting the captured media within hours vice weeks will allow for an increase in operating tempo and improved actionable intelligence. These benefits can be achieved through an automated analysis and reporting framework for media exploitation in computer forensics.

Metadata available from files provides a target area for investigators to concentrate on to obtain information about the file or possibly even determining who owns the file, or has been in contact with the file. For intelligence and law enforcement communities, examining metadata can potentially save substantial time frequently tied to manual analysis.

Metadata is data that describes data. To use an analogy from publishing, the content of a book can be thought of its *data*, and the book's title, publication year, author, page count, and other information is the book's metadata. In a digital forensics context, metadata available for examination frequently includes the file creator, creation times, modification times, titles, and number of revisions.

A. PURPOSE OF STUDY

This thesis presents a new technique for batch processing disk images and automatically extracting metadata from files and file contents. The technique is embodied in a program called `fiwalk` that has a plug-in architecture allowing new metadata extractors to be readily incorporated. `Fiwalk` is a program designed to retrieve information from disk partitions found on disk images. `Fiwalk` was chosen for use in this

thesis because `fiwalk` is also a metadata extraction system. Output from the program can be used directly by the `WEKA` data mining toolkit [52], or inserted into a database using the Structured Query Language (SQL).

This thesis seeks to answer the following questions regarding automated metadata extraction:

1. Can existing metadata extraction tools be combined for the automated processing of disk images?
2. In a forensics context, what metadata extraction opportunities exist within various file formats?
3. How does the metadata available for extraction between similar file format types compare?
4. Do the existing open source metadata extraction tools provide accurate results?

B. THESIS ORGANIZATION

Although there is limited work directly associated with metadata extraction, many studies and projects touch the effort. Chapter II discusses what metadata is available for file systems, media file formats, and document file formats. Chapter III presents a framework for automated metadata extraction and discusses the `fiwalk` program, the `Sleuth Kit`, and `ARFF` (attribute relation file format). Chapter IV describes the plug-ins for extracting metadata for `jpeg`, Microsoft Office files, along with a default metadata extractor built from `libextractor`. Chapter V presents a detailed look at the Microsoft Office 2007 document structure and compares Microsoft Office 2007 to Open Office, Chapter VI covers prior and related work, and Chapter VII presents findings, conclusions and suggests areas for future work.

II. METADATA

Metadata provides a significant amount of information about a file without examining the content of the file and serves a variety of purposes. For instance, metadata can improve search efficiency and the quality of the search results [22], can provide information about a digital photograph, is becoming more prevalent in legal cases [37], and on a more technical level, can be used to retrieve deleted files from a file system [10].

Metadata can be created in different ways. Some applications, such as Microsoft Office, automatically generate metadata when a file is created. Digital cameras and image manipulation programs also add metadata to image files [26]. Additionally, external content management and email systems automatically add metadata to files to aid in the tracking of the files [36]. Some applications allow the users to manually add metadata to their files. Apple provides the application `GarageBand`, which enables users to record, arrange, and mix music. Additionally, `GarageBand` provides a means for file owners to add metadata which can then be used by iTunes for cataloging and searching [4].

Metadata can be stored in different locations within files. Within Open Document Format (`ODF`) files, explicit metadata can be found in the `meta.xml` file [28]. `ODF` is an open and license-free file format released by OpenOffice.org as an alternative to the document file formats developed by Microsoft. In the predominantly used markup language for web pages `HTML` (Hypertext Markup Language), metadata can be tagged with the `<meta>` tag [22], and in Adobe's Portable Document Format (`PDF`) files, metadata can be stored in a document information dictionary or in a metadata stream [1].

From a forensics perspective, metadata can assist investigators with filename searches, timeline analysis, reporting, and reducing the number of files that need to be subjected to detailed analysis [32]. Metadata found on a hard drive can also be used to correlate ownership of the hard drive to a potential individual owner [20], and metadata within a file can be used to corroborate information about the file itself [26].

Two categories of metadata that are interesting to computer forensics analysts are file system metadata and digital media metadata. The remainder of this chapter discusses some types of metadata available in file systems and media files.

A. FILE SYSTEM METADATA

A distinction should be made between file system metadata files, which house the file system's administrative data, and file metadata, which contains information about the file's contents but is not actually the content of the file [10].

File system metadata usually contains metadata such as the specific data units allocated to a file, the size of the file, and access date/time stamps for the file. The exact metadata maintained depends on the file system (e.g., ext2/ext3, FAT, NTFS, UFS), and the data structure used by that file system [10].

File system metadata includes but is not limited to file owner, filenames, file sizes, MAC times, MD5 hashes, group ownership, and permissions [32]. In a Windows environment, by right-clicking on a file and selecting Properties, file metadata can be obtained. The metadata available includes the name of the file, the file type, the program the file opens with, the location on the disk, the size of the file, the size on the disk, and the created, modified, and accessed date/times.

Forensics applications are frequently concerned with extracting or retrieving files from disk images and with recreating timelines of system activity, which is why the usual metadata data definition is expanded to include where the file is found on the disk, fragmentation patterns, and MD5 hashes.

B. METADATA IN MEDIA FILES

Media files are prominent sources of metadata. This chapter examines the media file formats `GIF`, `JPEG`, `MP3/AAC`, and `TIFF` along with the metadata associated with these formats. Metadata associated with media files is important for forensics because this metadata provides investigators with potentially important evidence such as camera manufacturer, serial numbers, and file owners.

1. GIF

Graphics Interchange Format (GIF) defines a protocol designed for the on-line transfer and exchange of raster graphic data that is independent of the hardware used in their creation or display. GIF is defined in terms blocks and sub-blocks. The blocks and sub-blocks contain parameters and data utilized in the production of a graphic. A GIF data stream is a sequence of protocol blocks and sub-blocks representing a collection of graphics. Blocks can be divided into three categories: Control, Graphics-Rendering, and Special Purpose. Extensions can be found within the Special Purpose category [13].

GIF was invented by CompuServe in the 1980s [13] and was the first widely-used compressed image file format on the Internet. As a result, support for GIF has been in web browsers from the inception of the World Wide Web and still remains a very popular file format on the Internet today.

Unfortunately, GIF files possess limited metadata. However, one of the most lucrative metadata targets within a GIF file is the comment extension, an optional special purpose block that contains textual information not included as part of the actual graphics within a data stream. Comment extensions are typically used for credits, descriptions, or other types of non-control and non-graphical data. The recommended positioning of the comment extension within a data stream is at the beginning or the end of the stream [13].

A second meaningful location for metadata within a GIF file is the application extension. Like the comment extension, the application extension is an optional special purpose block within the data stream. The application extension contains application-specific information for specific programs to act upon. The extension can be for any purpose, which is why only special programs will recognize and act upon the information [13].

Once extracted, the comment extension may provide insight to an investigator regarding the origin or possibly the purpose of the image. Depending on the depth of the comments, the file owner or an associate of the owner could be identified.

Similarly, because the application extension reveals what application used the file, investigators may be able to construct a tighter connection between the image, and the drive where the image was found.

The open source metadata extraction tool, `libextractor`, provides a means for extracting metadata from `GIF` files [23].

2. JPEG

`JPEG` is the most popular file format today for the representation of digital photographs. Most if not all digital cameras directly support the `jpeg` format. `Jpegs` are widely distributed on the World Wide Web, and `jpeg` images are rich in metadata. One `jpeg` metadata standard is the Exchangeable Image File (EXIF) specification, created by the Japan Electronics and Information Technology Industries Association (JEITA). Metadata segments defined by the International Press Telecommunications Council (IPTC) and XMP from Adobe Systems are also popular [26].

The metadata within a `jpeg` file can include

- Make, model and serial number of the digital camera that took the photograph
- Date and time picture was taken
- Distance setting for the camera's focus
- Location information where the picture was taken (from a GPS)
- Thumbnail image of the picture [26]

Recognizing the types of information contained within the metadata of a `jpeg` file is important for many forensic applications. For example, many digital photograph software applications do not adjust the thumbnail image within the metadata when the primary picture is edited. If a picture is taken of a subject in a somewhat compromising position, the subject may not be aware that even though the compromising pose has been cropped out, the thumbnail image of the pose is likely still intact [26].

The location information from where a picture was taken can aid investigators that are piecing together a criminal puzzle. If investigators recover a camera from a crime

scene, and the images contain further evidence of a crime, being able to tie the images to a location can provide the investigators with additional leads. Additionally, with information on the "distance the camera was focus at," given the location of the photographed subject, it may be possible to pin point the exact position of the photographer [26]. Further, adding the date and time information into the pool of evidence can tighten the timeline for the investigators.

3. Music: MP3 and AAC

Advanced Audio Coding (AAC) and Moving Picture Expert Group – 1 (MPEG-1), audio layer 3 (MP3) represent compressed, lossy, perceptual coding scheme formats that are part of the MPEG set of standards for music encoding. AAC was enhanced for the MPEG-4 standard as MPEG-4 AAC [2].

MPEG-4 audio assists with a collection of applications that range from speech to high-quality multi-channel audio, and from natural to synthesized sounds. The primary benefit of MPEG-4 AAC over MP3 audio is that the clarity of MPEG-4 is approximately twice that of MP3 for similar bit rates, and files half the size for the same perceived quality [3].

Both AAC and MP3 formats include metadata specifications; however, standalone MP3 files do not contain a standard way of storing the metadata. In 1996 Eric Kemp developed a simple approach for adding a small chunk of data to the end of the audio file. The standard for storing this metadata was called ID3v1 for "IDentify an MP3". The ID3v1 tag occupies the last 128 bytes of an MP3 file and begins with the string "TAG". The tag allocates 30 bytes for the title, artist, album, and a "comment," 4 bytes for the year, and 1 byte as a predefined genre identifier placed at the end of the file [40,46].

ID3v1 tags were considered too short to contain enough meaningful metadata, so in 1998 the ID3v2 standard was introduced. Each ID3v2 tag holds one or more frames, which can be up to 16 MB in size for a total of 256 MB per tag. Each frame can contain any type of information such as album, title, artist, website, lyrics, equalizer presets,

copyright information, media type, and pictures. Essentially ID3v2 is another container specification. ID3v2 also provides some flexibility for including user added metadata [30].

Although AAC files also include metadata, the information is not contained within an ID3 tag [30]. Apple instead uses a proprietary audio file format referred to as the Apple Core Audio Format for audio data. The files under the Core Audio Format are chunk-based and contain AAC data formats [3]. Specifically, Apple uses Protected AAC to encode copy-protected music titles purchased from the iTunes Music Store [4].

The files purchased from the iTunes Music Store include the following metadata.

- Name
- Email address of purchaser
- Year
- Album
- Grouping
- Comments
- Genre
- Lyrics
- Artwork [4]

The metadata is useful for several reasons. First, the iTunes and iPod user interface is built from the metadata. Next, the metadata improves the efficiency of browsing and searching for files. Finally, the metadata serves to support and reinforce the content [4].

From a forensics standpoint, the metadata can be useful in identifying the owner or associating the files to a potential owner. The embedding of the email address provides one link, and user added comments or pictures could provide another link.

There a few tools available for extracting metadata from the ID3 tagged MP3 files and the AAC files. One tool is the MP3::Tag module, which is one of the Perl ID3 tag parsers that are available on CPAN (Comprehensive Perl Archive Network) [5]. MP3:: Perl Tag reads both ID3v1 and ID3v2 tags. Additionally, MP3::Tag allows the user to

edit the content of an MP3 file tag or create a new tag [46]. Another tool that is capable of extracting metadata from MP3 files is libextractor [23].

For AAC files, the MPEG4IP application provides numerous tools for working with audio and visual files including files in AAC format. Specifically, the mp4dump is a utility that can extract mp4 file metadata into a text form. The MPEG4IP application also includes an mp4info utility that provides mp4 file summary information [36].

4. Tagged Image File Format (TIFF)

Tagged Image File Format (TIFF) files are container files that can hold data and metadata in a variety of formats. Information that can be included in a TIFF version 6.0 file includes:

- The date and time that the image was created.
- An image description
- Make and model of the equipment that produced the image
- Software version that produced the image
- Creator of the image (artist)
- Copyright holder [31]

C. METADATA IN DOCUMENT FILES

1. Microsoft Office

Microsoft Office documents are some of the most popular user generated documents in existence today. Some common examples of Microsoft Office documents include Microsoft Word, Excel, and Power Point documents. For Microsoft Office document versions 1995 – 2003, Microsoft used Object Linking and Embedding (OLE) technology to manage its file format. OLE is technology that allows applications to create compound documents from multiple sources [39]. With Microsoft Office 2007, Microsoft began using the Office Open XML format.

Microsoft Office documents contain a variety of metadata. By simply opening a Microsoft Word document, and selecting properties from the File menu, metadata such as

author, title, company, subject, comments, keywords, modified, accessed, and created times, and file location can easily be found. Additionally, in some documents, information can be found regarding document reviewers and savers [47].

Microsoft Office documents also contain hidden information, data that is not available through the Office application's interfaces, or that can be hidden from view using application settings. Examples of hidden data include track changes and comments, which can be hidden from view as a default setting, and author history and fast save data, which are not accessible from the native application interfaces [47].

Several stories have surfaced regarding hidden information in Microsoft Word documents. One of the most famous involved a UK Government issued document regarding Iraq's concealment of weapons of mass destruction. The Word version of the document was placed on the Internet, and when the document was analyzed, the contents were discovered to be largely based on a journal article written by a postgraduate student in the United States. A deeper look into the document's revision log revealed the names of four of the people, who prepared the document for publication along with the offices where some of them worked [51].

Computer forensics investigators can also obtain valuable information or case leads from metadata and hidden data within Microsoft Office documents. The BTK killer was caught after police were able to extract the metadata containing his name and the name of the church housing the computer he used to create a Microsoft Word document he sent to the police [42].

There are a few tools available for extracting metadata from Microsoft OLE formatted files. One is the `wv` library. The `wv` library can load and parse Microsoft Word 2003, 2000, 97, 95 and 6 file formats. Included with `wv` is an application called `wvWare`, a command-line driven application with helper scripts–one of which is used for extracting metadata from files. `WvSummary` is a useful script that prints out the metadata from Microsoft Office documents [33].

Microsoft also provides Visual C++ code that extracts metadata from office documents and produces the output found in Figure 1 [27]. However, this code requires

the use of proprietary Microsoft DLLs, and will, as a result, only run on Windows. Since most digital forensic tools are written and run on Unix-based platforms, many investigators would probably choose to not use the Visual C++ code because of the Windows environment limitation.

```
Title:    MyTitle (VT_LPSTR)
Subject:  MySubject (VT_LPSTR)
Author:   MyAuthor (VT_LPSTR)
Keywords: MyKeywords (VT_LPSTR)
Comments: MyComments (VT_LPSTR)
Template: Normal (VT_LPSTR)
LastAuthor: Me (VT_LPSTR)

Revision Number: 8 (VT_LPSTR)

Edit Time: 01:05.47 pm, Mon 01/01/1601 (VT_FILETIME)
Last printed: (VT_EMPTY)
Created: 01:42.00 pm, Fri 05/29/1998 (VT_FILETIME)
Last Saved: 12:31.00 pm, Mon 06/01/1998 (VT_FILETIME)
Page Count: 1 (VT_I4)
Word Count: 3(VT_I4)
Char Count: 19 (VT_I4)
 Thumbnail: (VT_EMPTY)
   AppName: Microsoft Word 8.0 (VT_LPSTR)
Doc Security: 0 (VT_I4) [27]
```

Figure 1. Visual C++ metadata extract

2. Office Open XML Format (Microsoft Office 2007)

With the Microsoft Office 2007 release, Microsoft began using its Office Open XML format. Microsoft contends that the new formats improve file and data management, data recovery, and interoperability with line-of-business systems. Under the new format, any application that supports XML can access and work with data in the new format [45]. This section provides an introductory look at the new Office Open format. A more detailed analysis is conducted in Chapter V. Table 1 below presents a list of the different Office Open XML file types and their extensions [45].

The new file format container is based on the ZIP compressed file format specification (Figure 2). The document parts are stored in the file container and are mostly comprised of XML files that describe document properties, application data, metadata, and customer data [45].

Document Format	File Extension
Word 2007 XML Document	*.docx
Word 2007 XML Macro-Enabled Document	*.docm
Word 2007 XML Template	*.dotx
Word 2007 XML Macro-Enabled Template	*.dotm
Excel 2007 XML Workbook	*.xlsx
Excel 2007 XML Macro-Enabled Workbook	*.xlsm
Excel 2007 XML Template	*.xltx
Excel 2007 XML Macro-Enabled Template	*.xltm
Excel 2007 XML Binary Workbook	*.xlsb
Excel 2007 XML Macro-Enabled Add-In	*.xlam
PowerPoint 2007 XML Presentation	*.pptx
PowerPoint 2007 Macro-Enabled XML Presentation	*.pptm
PowerPoint 2007 XML Template	*.potx
PowerPoint 2007 Macro-Enabled XML Template	*.potm
PowerPoint 2007 Macro-Enabled Add-In	*.ppam
PowerPoint 2007 XML Show	*.ppsx
PowerPoint 2007 Macro-Enabled XML Show	*.ppsm

Table 1. Office Open XML file types

```
Archive:  word1-save1.docx
  Length      Date    Time    Name
 --------    ----    ----    ----
     1364  01-01-80 00:00   [Content_Types].xml
      735  01-01-80 00:00   _rels/.rels
      817  01-01-80 00:00   word/_rels/document.xml.rels
     1062  01-01-80 00:00   word/document.xml
     7559  01-01-80 00:00   word/theme/theme1.xml
    12840  01-01-80 00:00   docProps/thumbnail.jpeg
     1963  01-01-80 00:00   word/settings.xml
     1521  01-01-80 00:00   word/fontTable.xml
      276  01-01-80 00:00   word/webSettings.xml
      710  01-01-80 00:00   docProps/core.xml
    15019  01-01-80 00:00   word/styles.xml
      734  01-01-80 00:00   docProps/app.xml
 --------                   -------
    44600                   12 files
```

Figure 2. Example Microsoft Word 2008 file directory (Macintosh)

For example the Document Properties Part contains the core document properties defined for all files conforming to the XPS format. Some of the properties include the author, title, subject, comments, last saved date, and created date [45]. Figure 3 is a sanitized excerpt from the `core.xml` file found in the `docprops` folder.

```
<dc:title />
  <dc:subject />
  <dc:creator>authorFirst authorLast</dc:creator>
  <cp:keywords />
  <dc:description />
  <cp:lastModifiedBy>authorFirst authorLast</cp:lastModifiedBy>
  <cp:revision>1</cp:revision>
  <dcterms:created xsi:type="dcterms:W3CDTF">2008-01-
   24T21:42:00Z</dcterms:created>
  <dcterms:modified xsi:type="dcterms:W3CDTF">2008-01-
   24T21:42:00Z</dcterms:modified>
  </cp:coreProperties>
```

Figure 3. Example from core.xml file

There are several interesting files and folders that comprise the various component parts of an Office Open XML document. A brief description of each is presented below.

- `Application Folder` – contains the application specific document component files for instance Excel, Word, Power Point.
- `Media Folder` – contains the `image.jpeg` files that were embedded within a document as picture content controls.
- `Audio File` – contains any audio files such as `.mid`, `.mp3`, or `.wav`. These files potentially contain their own metadata and could be interesting to an investigator on their own accord.
- `_rels Folder` - contains a `.rels` file that defines the root relationship within the package. The `.rels` file within this folder contains a unique relationship Id.
- `document.xml` file - an `XML` file that contains the main text and body of the document [45]

In addition to the common metadata such as author, creation date, modified date, revision number, and thumbnail image filename found in the files and folders listed above, users can add their own data and content by creating custom-defined xml and placing it in the file as another part [45].

Within a `.docx` document, there is also a Revision Identifier for a paragraph (rsidR) whose values are used to identify the editing session in which a paragraph was added to a main document. An `rsidR` value should be unique, and instances with the same value within a single document indicate that the regions were modified during the

same editing session. Associated with an `rsidR` attribute is the `rsidRDefault` attribute. An `rsidRDefault` attribute is used for all runs within a paragraph in which an `rsidR` is not defined [14].

Overall, Office 2007 documents contain a significant amount of information that can be extracted from a `.docx` package beyond the standard author, creation/modification dates, that were obtained from prior versions of Office, and the Office Open XML format provides investigators with a simpler environment in which they can extract desired metadata from the xml files. For instance, a simple routine could be written to create a list of documents written by a specific individual, or a list of comments extracted from the xml files [45]. Additionally, the inclusion of a `thumbnail.jpeg` file or other embedded image within the package provides investigators with additional metadata which can be obtained by running the `Exif` program on the file. The metadata can then be analyzed by investigators. Finally by examining `rsid` values within the XML, investigators can identify documents that were created during the same editing session, but later dispersed and modified separately.

3. Open Office

A similar file structure to the Office Open XML format is the Open Document Format Alliance (ODF) which was created to facilitate sharing of information between different word processing applications. Like Office Open XML files, an ODF file is basically a zipped archive comprised of several XML files. The exact files and directories contained within an ODF file will differ depending on the systems the document was created on and the type of information stored within the ODF file itself. Table 2 provides a list of different ODF file types and their extensions [28].

When an ODF text file is unzipped, the archive will have some of the following files:

- mimetype
- content.xml
- styles.xml
- meta.xml

- settings.xml
- META-INF/manifest.xml [28]

Document Format	File Extension
OpenDocument Text	*.odt
OpenDocument Text Template	*.ott
OpenDocument Master Document	*.odm
HTML Document	*.html
HTML Document Template	*.oth
OpenDocument Spreadsheet	*.ods
OpenDocument Spreadsheet Template	*.ots
OpenDocument Drawing	*.odg
OpenDocument Drawing Template	*.otg
OpenDocument Presentation	*.odp
OpenDocument Presentation Template	*.otp
OpenDocument Formula	*.odf
OpenDocument Database	*odb

Table 2. ODF File Types

Of these files, `mimetype`, `meta.xml`, `META-INF/manifest.xml`, and `content.xml` offer the most potential for extracting meaningful metadata. The mimetype file is a single line file with the `mimetype` of the content file. `META-INF/manifest.xml` contains a listing of all the files in the zipped archive, and `meta.xml` holds the metadata such as the author and creation dates. Although not necessarily a source of metadata, the `content.xml` file should be identified because this file contains the data for the document itself [28].

The potential uses of metadata associated with Open Office documents follow the same path as the uses of metadata with Microsoft Office 2007 documents. Investigators can readily search for all files created by the same author, extract files based on creation or modification dates to build timelines, and they can generate associations between files with the same authors appearing on different drives. Additionally, the `manifest.xml` file provides a resource for investigators to use when accounting for all files in the archive. If an investigator notices that a file is listed in the `manifest.xml` file, but is not included in the archive directory structure, the investigator now has a lead to search for a potentially interesting file that has been deleted and should be recovered.

We have written a tool in Python called odf_extractor to extract metadata from Open Document Format files.

4. Portable Document Format (PDF)

Adobe PDF is the organic file format used by the Adobe family of products. The primary goal of the PDF is to allow users to easily exchange and view unmodifiable electronic documents. A PDF file may contain metadata such as title, author, creation date, and modification date [1]. Metadata within a PDF file can be stored in one of two ways:

- In a document information dictionary
- In a metadata stream [1]

Within the trailer of a PDF file, there is an optional Info entry that holds the document information dictionary containing metadata for the document. The contents of the document information dictionary are as follows:

- Title - document's title (optional)
- Author - name of the person who created the document (optional)
- Subject - subject of the document (optional)
- Keywords - keywords associated with the document (optional)
- Creator – name of application that was used to originally create the document if document was converted to PDF (optional)
- Producer – name of application used to convert document to PDF from another format if a conversion took place (optional)
- CreationDate - date and time the document was created (optional)
- ModDate - date and time the document was most recently modified (optional)
- Trapped - indicates whether document was modified to include trapping information (optional)
- Values:
    - True - document has been fully trapped; no further trapping is needed.
    - False - document has not yet been trapped; any desired trapping must still be done.

- Unknown - either it is unknown whether the document has been trapped, or it has been partially but not yet fully trapped. Additional trapping may still be needed (Default) [1].

Figure 4 shows an example of a typical document information dictionary.

```
1 0 obj
<< /Title (PostScript Language Reference, Third Edition)
/Author (Adobe Systems Incorporated)
/Creator (Adobe® FrameMaker® 5.5.3 for Power Macintosh)
/Producer (Acrobat® Distiller™ 3.01 for Power Macintosh)
/CreationDate (D:19970915110347-08'00')
/ModDate (D:19990209153925-08'00')
>>
endobj [1]
```

Figure 4. Document Information Dictionary

The second way to store metadata within a PDF file is in a metadata stream. Metadata streams hold two primary advantages over document information dictionaries.

First, metadata-bearing artwork is frequently embedded as a component within larger documents by PDF-based workflows, so metadata streams standardize the preservation of these components for future examination [1].

Second, because PDF documents are usually available on the Internet or other environment, the documents should be easily examined, catalogued, and classified by the many tools used to perform these functions. The tools should be able to comprehend the self-contained description of the document even if the tools do not understand PDF [1].

XML is used to represent the contents of a metadata stream. The XML is visible as plain text only if the tools are PDF aware, or if the tools are not PDF aware if the metadata stream is both unfiltered and unencrypted. The specific format of XML used is defined as part of the Extensible Markup Platform framework [1].

The primary function of metadata within the PDF document is to facilitate cataloguing and searching for documents in external databases [1]. The metadata retrieved from PDF files offers investigators many of the same benefits as metadata retrieved from Microsoft Office and Open Office documents. One point of interest is that many users will “convert” their Microsoft Office documents to a PDF format to eliminate

the possibility of disclosing hidden information. In his article, Ward quoted David Stevenson of Adobe, who confirmed that PDF files were usually the final version of a document, and typically the revision history of the document was concealed. The information available depends on the authoring software used to create the document [51].

Libextractor is one of the tools currently available for extracting metadata from PDF files [23].

III. A FRAMEWORK FOR AUTOMATED METADATA EXTRACTION

A. FIWALK

1. fiwalk Introduction

File and Inode Walk (`fiwalk`) is an application that retrieves information from disk partitions found on disk images. `Fiwalk` relies on the programmer's interface from Brian Carrier's popular open source digital forensics tool, `Sleuthkit` (TSK) to locate all of the files and orphaned inodes found in a given disk image [11]. The application can optionally compute the MD5 or SHA1 cryptographic hash of any objects and then save the objects.

`Fiwalk` is also a metadata extraction system. The file system metadata is obtained through TSK, and the document metadata is retrieved by using the various metadata extraction plug-ins. A disk image is fed into `fiwalk`, and metadata from the image is output formatted as an ARFF (attribute-relation file format) file or a "walk file". The user can make the determination as to which format the output should be in. Figure 5 provides the output options.

```
Output options:
    -A = ARFF output
    -X = XML output (not currently implemented)
    -B = output 512-byte bloom filter
    -d = debug this program
    -v = Enable SleuthKit verbose flag
```

Figure 5. fiwalk output options

2. fiwalk Algorithm

`Fiwalk` is built around the basic algorithm provided below.

```
1 - Find all of the partitions on the disk.
2 - For each partition, walk the files.
3 - For each file, print the requested information.
4 - For each partition, walk the inodes
5 - For each inode, print the requested information.
```

First, `fiwalk` obtains the file system and file metadata by using the TSK programmer's interface. The next subchapter provides a more detailed examination of the libraries offered through TSK.

Next `fiwalk` obtains a list of all the partitions, and for each partition, `fiwalk` derives a list of all the files. For each file, the file system and associated metadata is obtained from TSK, and the appropriate plug-in is called to extract the metadata. Once the file is "processed," an ARFF record is created.

The user can also choose to process the orphaned files; concurrently, an ARFF record is generated for the orphaned file, but the plug-ins are not invoked during orphan file processing because the plug-in system relies on filenames to figure out which plug-in to invoke.

3. fiwalk Evaluation

To evaluate `fiwalk`, disk images obtained from sources in India, Mexico, China, Israel, as well as a control image were processed through `fiwalk`. Figure 6 below provides an excerpt of a disk image that was processed using `fiwalk` and output in ARFF.

```
3627, 1, 78, "2007-11-06 19:40:24", "2007-11-06 19:39:56", "2008-02-22
08:00:00", 1, 2637992, ?, "Documents and Settings/jjm2007/My
Documents/desktop.ini", jjm2007
3628, 1, 9974, "2008-01-25 04:20:02", "2008-01-25 04:20:02", "2008-02-
22 08:00:00", 1, 2638008, ?, "Documents and Settings/jjm2007/My
Documents/DOCX Files/Hello World.docx", jjm2007
3629, 1, 15066, "2008-02-06 05:51:16", "2008-02-06 05:51:16", "2008-02-
22 08:00:00", 1, 2638032, ?, "Documents and Settings/jjm2007/My
Documents/DOCX Files/Hello_World2.zip", jjm2007
3630, 1, 2025, "1980-01-01 08:00:00", "1980-01-01 08:00:00", "2008-02-
22 08:00:00", 1, 2638080, ?, "Documents and Settings/jjm2007/My
Documents/DOCX
Files/MS2007_Feb_13_08/MS2007_Feb_13_08/[Content_Types].xml", jjm2007
```

Figure 6. Output from fiwalk in ARFF

B. USING THE SLEUTHKIT PROGRAMMATICALLY

The most popular open source forensics tool is `Sleuthkit` and `Autopsy`, maintained by Brain Carrier.

The `Sleuthkit` (TSK) and `Autopsy` forensic browser are Unix-based tools that were initially released in 2001. TSK is composite of over 20 command-line tools used for analyzing disk and file system images for evidence. Autopsy is a front-end browser for TSK created to make the analysis process easier for the user [10].

In addition to TSK's capability for analyzing disk images, one of its primary strengths is the libraries available for use by forensic tool developers.

TSK consists of four logical libraries. The first library is the disk image file format library. This library provides an abstraction to the various file formats such as Expert Witness format, Advanced Forensic Format (`AFF`), and raw format. The image file format library presents a read-only interface to disk image files and allows programs to read data from arbitrary locations in the file without needing to know what format is being utilized [11].

The next library is the volume system (media management) library, which analyzes the various types of partitions that TSK supports. The volume system library has two major interfaces. The first is a function to open a disk image file and detect the volume system type. The second is a "walk" function that identifies the volume on a disk and processes a callback function for each volume.

The third library is for file system tools and is the largest. The design is similar to the volume system library in that a program opens the disk or partition image file and then accesses "walk" functions at the data, metadata, filename, file content, and journal levels.

The final library is the file system library which interprets file system structures and allows the retrieval of directory and file system information.

Although these libraries are described separately, in the current version of TSK, they are distributed as a single library, `libtsk` [11].

`Fiwalk` relies on the TSK interfaces listed below:

`FS_INFO` – structure that describes file systems
`IMG_INFO` – structure that describes disk images
`img_open` - opens the image; calls `do_dimage`
`do_dimage` - analyzes the image
`mm_open` – opens the volume; `mm_part_walk`
  - walk each volume;
  - calls `mm_act` for each
`mm_act` - analyzes each partition; calls do_vol for ones we can analyze
`do_vol` - analyzes each volume
`fs_open` - opens the file system; `dent_walk`
  - walks the directory structure.
  - calls `dent_act` for each file
`dent_act` - prints file name; calls `file_walk()` to read each file;
  - calls `file_act` for each sector
`file_act` - prints the block number

C. DOMEX-GATEWAY INTERFACE (DGI)

DGI or Domex-Gateway Interface is the means by which plug-ins communicate with `fiwalk`. `Fiwalk` puts the data into a file in the file system; the plug-in reads the file and returns the data as a series of name:value pairs. Some of the name:value pairs come from TSK while others come from the plug-ins. The job of `fiwalk` is to collect all of the name:value pairs from each file, as provided by TSK, augment them with name:value pairs from the plug-ins, and place the result into a single ARFF file.

D. ARFF

When working on a data mining problem, bringing the data together into a set of instances is the first step. Integrating data from different sources presents many challenges such as differing styles, conventions, time periods, degrees of aggregation, and so on. By standardizing the format of the data, many of these problems can be alleviated. The attribute-relation file format (ARFF) provides a standard way of representing data sets that consist of independent, unordered instances without involving relationships between the instances. The machine learning software `Weka` accepts ARFF files as a standard input format [52].

In order to support a standardized format for analyzing data, `fiwalk` provides a user option for producing requested data from a disk image in ARFF.

IV. PLUG-INS FOR AUTOMATED METADATA EXTRACTION

A. JPEG PLUG-IN (JPEG_EXTRACT)

As previously identified, `jpeg` is the most popular file type today for representing digital images. Being able to extract the metadata from the images is important for investigators. One of the best known extractor programs of metadata in JPEG files is the `exif` program [34]. The `fiwalk` program comes with a plug-in that uses the `exif` program called `jpeg_extract`.

If a JPEG file is encountered, `jpeg_extract` calls the `exif` program to extract the metadata. The metadata returned by exif is processed by `jpeg_extract` and placed it into DGI format for further processing/exporting.

Figure 7 below is a sample run of `exif` followed by a sample run (Figure 8) from `jpeg_extract` using the same file. Output has been truncated for brevity.

```
Manufacturer      Canon
Model Canon EOS DIGITAL REBEL XTi
Orientation top - left
Date and Time     2007:12:23 00:53:23
ThumbnailSize     9822
…
```

Figure 7. Output from exif program

```
Manufacturer: Canon
Model: Canon EOS DIGITAL REBEL XTi
Orientation: top - left
Date-and-Time: 2007:12:23 00:53:23
ThumbnailSize: 9822
…
```

Figure 8. Output from jpeg_extract in DGI format

B. MICROSOFT OFFICE PLUG-IN

To extract metadata from Microsoft Office documents, two metadata extraction plug-ins are implemented. The first plug-in utilizes `wvSummary`, a previously mentioned "helper script" for `wvWare`. The second plug-in is an expanded XML parser written in

Python. This plug-in used to extract metadata from Office Open XML files, the native format of Microsoft Office 2007 (Windows), and 2008 (Macintosh).

1. WV PLUG-IN (Word_Extract)

If a Microsoft Office document (other than Microsoft Office 2007) is recognized, then `word_extract.java` is used. `Word_extract` is similar to the `jpeg_extract` plug-in, and converts the `wvSummary` output into DGI format. Figure 9 shows the results from `wvSummary` after the output is translated into DGI format.

```
Filename: samples/LibExtractTestFiles/TestFileWithImages.doc
Editing-Duration: 2009-04-22T19:30:48Z
msole-codepage: 1252
Generator:  Microsoft Word 10.0
Last-Modified: 2008-03-04T03:55:00Z
Creator:  James and Diane Migletz
Revision:  3
Number-of-Pages: 1
Number-of-Words: 871
Title:  Task analysis for using Play Station 2 to play DVD movie
Created: 2008-03-04T03:50:00Z
Subject:
Template:  Normal.dot
Keywords:  Thesis, test
Description:
Number-of-Characters: 4971
Security-Level: 4971
Last-Saved-by:  Diane Migletz
Received-From:  James
msole-codepage: -535
Number-of-Lines: 41
Document-Parts: [0, Task analysis for using Play Station 2 to play DVD movie)]
Default-Locale: 1033
Number-of-Paragraphs: 11
Unknown1: 5831
Company:   Art Teacher
Scale: FALSE
Links-Dirty: FALSE
Unknown3: FALSE
Document-Pairs: [(0,  Title ), (1, 1)]
Editor:  First Test
Unknown6: FALSE
Unknown7: 662190
Checked-By:  Professor
```

Figure 9. Output of Microsoft Word file after processing by word_extract

Figures 10 and 11 contain the output produced by `word_extract` for Microsoft Excel and Power Point documents respectively. The results are very similar to what Microsoft Word displays when the `File -> Properties` menu option is selected.

```
Filename: samples/LibExtractTestFiles/TestSpreadsheet.xls
msole-codepage: 1252
Generator:  Microsoft Excel
Last-Modified: 2008-03-05T05:10:40Z
Creator:  Diane Migletz
Title:  Test Spreadsheet
Created: 2008-03-05T05:00:37Z
Subject:  wvSummary Test
Keywords:  Thesis, Metadata, test sheet
Description:  Test spreadsheet for the wvSummary plugin.
Security-Level: 1093994496
Last-Saved-by:  Diane Migletz
msole-codepage: 1252
Manager:  James Migletz
Document-Parts: [(0,  TestSheet1 ), (1,  TestSheet2 ), (2,  Sheet3 )]
Category:  School Related
Company:  Home
Scale: FALSE
Links-Dirty: FALSE
Unknown3: FALSE
Document-Pairs: [(0,  Worksheets ), (1, 1701144659)]
Unknown6: FALSE
Unknown7: 662190
```

Figure 10. Output of Microsoft Excel file after processing by word_extract

```
      Filename: samples/LibExtractTestFiles/TestPPPresentation.ppt
      Editing-Duration: 2009-04-22T19:27:12Z
      msole-codepage: 1252
      Generator:  Microsoft PowerPoint
      Last-Modified: 2008-03-05T05:15:00Z
      Creator:  Diane Migletz
      Revision:  3
      Number-of-Words: 18
      Title:  Test Power Point Presentation
      Created: 2008-03-05T05:07:19Z
      Subject:  Test PP presentation for wvSummary
      Keywords:  Thesis, metadata, Power Point
      Description:  Test presentation for the wvSummary plugin.
      Thumbnail: ((GsfClipData*) 0x9b451a0)
      Last-Saved-by:  Diane Migletz
      msole-codepage: 1252
      Number-of-Bytes-in-the-Document: 3502669
      Manager:  James Migletz
      Document-Parts: [(0,  Arial ), (1,  Default Design ), (2,  Test
Power Point Presentation ), (3,  Slide 2 ), (4,  Slide 3 ), (5,  Hidden
slide )]
      Category:  School Related
      Number-of-Slides: 4
      Number-of-Paragraphs: 7
      Number-of-Notes: 1
      Number-of-'Multi-Media'-Clips: 0
      :  On-screen Show
      Company:  Home
      Number-of-Hidden-Slides: 1
      Scale: FALSE
      Links-Dirty: FALSE
      Unknown3: FALSE
      Document-Pairs: [(0,  Fonts Used ), (1, 1), (2,  Design Template
), (3, 1), (4,  Slide Titles ), (5, 218116896)]
      Unknown6: FALSE
      Unknown7: 662190
```

Figure 11. Output of Microsoft PowerPoint file after processing by word_extract

2. DOCX_Extractor

When a Microsoft Office 2007 file in the Office Open XML format is encountered, `fiwalk` invokes the `docx_extractor` plug-in to retrieve metadata from Word, Excel, and Power Point documents.

The extractor begins by unzipping the `docx`, `xlsx`, or `pptx` archive. Next the extractor analyzes the XML to find values associated with alias and tag, id attributes of the structured document tag `<w:sdt>`. These values are important because they provide

additional information that can be used to identify content controls such as textboxes and picture content (`.jpeg`) files that have been added to a document. For instance, when a user adds a content control in Microsoft Word 2007, a data entry screen in displayed for the user to manually enter the *Title* and *Tag* names for the control. The alias value in the `<w:sdt>` node is tied to the *Title*, and the tag value in the `<w:sdt>` node is derived from the *Tag* entry.

The extractor also parses the XML to retrieve data store ids and GUIDs, which bind the custom pieces to the correct data within the data store [8].

The final phase of extraction within the program focuses on more traditional metadata. The extractor retrieves metadata found embedded in the following XML tags.

- Creator
- Last Modified
- Created date
- Modified date
- Title
- Subject
- Description
- Application
- Company
- Number of characters, words, lines, paragraphs, pages, etc.
- Template
- Revision number

Figure 12 contains the output of a `.docx` document after being processed by docx_extractor. Note that different `sdtid` numbers are assigned for the different content controls, along with different default paragraph revision id numbers, and a single `GUID` (globally unique identifier) are all present within the `.docx` archive. Figure 13 is the output of a `.xlsx` document, and Figure 14 is the out put of a `.pptx` document after processing by docx_extractor. A limitation of ARFF is that the format does not support one-to-many relationships, so unique “names” (Archive-File1, Content-Control2, etc.) for the name:value pairs are necessary to address this limitation.

```
Archive-File1: docProps/app.xml
Archive-File2: docProps/core.xml
Archive-File3: word/document.xml
Archive-File4: webSettings.xml
Archive-File5: settings.xml
Archive-File6: styles.xml
Archive-File7: theme/theme1.xml
Archive-File8: glossary/document.xml
Archive-File9: fontTable.xml
Paragraph-Revision-ID1: 000F1AD8
Paragraph-Revision-ID-Default1: 00384658
Content-Control-Alias1: This tests a plain text content control
Content-Control-Alias2: This tests a combo box
Content-Control1: PlainText1
Content-Control2: combo1
Content-Control-Id1: 12541641
Content-Control-Id2: 12541644
Paragraph-Revision-ID2: 00000000
Paragraph-Revision-ID-Default2: 002242F4
GUID1: {594B704E-F2DF-432A-86C3-8AC42839A87D}
GUID2: {7313B4A8-96F3-436E-83A8-C31E3D34AF0D}
Generator:  Microsoft Office Word
Company:  NPS
Template:  Normal.dotm
Number-of-Pages:  1
Number-of-Lines:  1
Number-of-Paragraphs:  1
Number-of-Words:  29
Number-of-Characters:  166
Created:  2008-02-13T17:35:00Z
Last-Modified:  2008-02-13T17:35:00Z
Creator:  James Migletz
Revision:  2
LastSavedBy:  James Migletz
```

Figure 12. Output of .docx document after processing by docx_extractor

```
Archive-File1: docProps/app.xml
Archive-File2: docProps/core.xml
Archive-File3: xl/workbook.xml
Archive-File4: worksheets/sheet3.xml
Archive-File5: worksheets/sheet2.xml
Archive-File6: worksheets/sheet1.xml
Archive-File7: styles.xml
Archive-File8: theme/theme1.xml
Created:  2008-02-07T20:06:09Z
Last-Modified:  2008-02-07T20:06:41Z
Creator:  James Migletz
LastSavedBy:  James Migletz
Generator:  Microsoft Excel
Company:  NPS
```

Figure 13. Output of .xlsx document after processing by docx_extractor

```
Archive-File1: docProps/core.xml
Archive-File2: docProps/thumbnail.jpeg
Archive-File3: ppt/presentation.xml
Archive-File4: docProps/app.xml
Archive-File5: ../slideLayouts/slideLayout1.xml
Archive-File6: presProps.xml
Archive-File7: slides/slide1.xml
Archive-File8: slideMasters/slideMaster1.xml
Archive-File9: tableStyles.xml
Archive-File10: theme/theme1.xml
Archive-File11: viewProps.xml
Archive-File12: ../slideMasters/slideMaster1.xml
Archive-File13: ../slideLayouts/slideLayout8.xml
Archive-File14: ../slideLayouts/slideLayout3.xml
Archive-File15: ../slideLayouts/slideLayout7.xml
Archive-File16: ../theme/theme1.xml
Archive-File17: ../slideLayouts/slideLayout2.xml
Archive-File18: ../slideLayouts/slideLayout6.xml
Archive-File19: ../slideLayouts/slideLayout11.xml
Archive-File20: ../slideLayouts/slideLayout5.xml
Archive-File21: ../slideLayouts/slideLayout10.xml
Archive-File22: ../slideLayouts/slideLayout4.xml
Archive-File23: ../slideLayouts/slideLayout9.xml
Created:  2008-02-07T20:09:53Z
Last-Modified:  2008-02-07T20:10:32Z
Creator:  James Migletz
Title:  Slide 1
Revision:  1
LastSavedBy:  James Migletz
Generator:  Microsoft Office PowerPoint
Company:  NPS
Number-of-Paragraphs:  0
Number-of-Words:  0
Number-of-Slides:  1
Number-of-Hidden-Slides:  0
Number-of-Notes:  0
Number-of-'Multi-Media'-Clips:  0
Presentation-Format:  On-screen Show (4:3)
```

Figure 14. Output of .pptx document after processing by docx_extractor

C. DEFAULT PLUG-IN (LIBEXTRACT_PLUGIN)

The default plug-in uses `libextractor`, a tool capable of reading metadata from a wide array of file formats [23].

`Libextractor` is similar to the Unix `file` program, which attempts to classify a given file based on the contents of the file as opposed to making a determination based solely on the file extension [35]. However, `libextractor` is different from `file` in

that libextractor attempts to derive more than just the mime type [23]. For instance, libextractor can identify additional information such as the name of the software used to create the file, the author, descriptions, album titles, image dimensions, or the length of a movie.

Libextractor retrieves the metadata by implementing a parser for the different formats. The current types of formats supported under libextractor include MP3, Ogg, Real Media, MPEG, RIFF (avi), GIF, JPEG, PNG, TIFF, HTML, PDF, PostScript, Zip, OpenOffice.org, StarOffice, Microsoft Office, tar, DVI, man, Deb, elf, RPM, and asf [23].

In general libextractor can handle a wider range of document types but does not cover any of them as deeply as specially written plug-ins. For this reason, an fiwalk plug-in for libextractor was created to be called when a more specific plug-in is not available.

Figure 15 is the resulting output of a .jpeg file after being processed by libextractor, and Figure 16 is the output for the same file after being run through jpeg_extract. Note that there is significantly more metadata obtained from jpeg_extract than from libextractor.

```
white balance - Auto
image quality - Fine
macro mode - (0)
metering mode - Average
exposure mode - Aperture priority
iso speed - 200
focal length - 120.0 mm
flash bias - 0 EV
flash - No
exposure bias - 0
aperture - F5.6
exposure - 1/180 s
date - 2005:10:22 11:50:27
orientation - left, bottom
camera model - Canon EOS 10D
camera make - Canon
size - 3072x2048
mimetype - image/jpeg
```

Figure 15. Output from libextractor

```
Manufacturer: Canon
Model: Canon EOS 10D
Orientation: left - bottom
x-Resolution: 180.00
y-Resolution: 180.00
Resolution-Unit: Inch
Date-and-Time: 2005:10:22 11:50:27
YCbCr-Positioning: centered
Compression: JPEG compression
x-Resolution: 180.00
y-Resolution: 180.00
Resolution-Unit: Inch
Exposure-Time: 1/179 sec.
FNumber: f/5.6
ISO-Speed-Ratings: 200
Exif-Version: Exif Version 2.2
Date-and-Time--original-: 2005:10:22 11:50:27
Date-and-Time--digitized-: 2005:10:22 11:50:27
ComponentsConfiguration: Y Cb Cr -
Compressed-Bits-per-Pixel: 3.00
Shutter-speed: 7.49 EV (APEX: 13, 1/179 sec.)
Aperture: 4.97 EV (f/5.6)
Exposure-Bias: 0.00 EV
MaxApertureValue: 2.97 EV (f/2.8)
Metering-Mode: Average
Flash: Flash did not fire.
Focal-Length: 120.0 mm
Maker-Note: 1372 bytes unknown data
User-Comment:
FlashPixVersion: FlashPix Version 1.0
Color-Space: sRGB
PixelXDimension: 3072
PixelYDimension: 2048
Focal-Plane-x-Resolution: 3443.95
Focal-Plane-y-Resolution: 3442.02
Focal-Plane-Resolution-Unit: Inch
Sensing-Method: One-chip color area sensor
File-Source: DSC
Custom-Rendered: Normal process
Exposure-Mode: Auto exposure
White-Balance: Auto white balance
Scene-Capture-Type: Standard
InteroperabilityIndex: R98
InteroperabilityVersion: 0100
RelatedImageWidth: 3072
RelatedImageLength: 2048
ThumbnailSize: 10240
```

Figure 16. Output from jpeg_extract

When a suitable file type such as `PDF`, `HTML`, `GIF`, etc. is encountered, the file is passed to `libextractor` through the plug-in. This program sends the file to `libextractor` and then converts the output into DGI format just as the `jpeg_extract` plug-in.

Figure 17 provides output generated from `libextractor` when a `PDF` file is used as input, and Figure 18 shows the same output when the file is processed through `Libextract_plugin.java`.

```
creation date - 20080106161535-08'00'
producer - OpenOffice.org 2.3
creator - Impress
format - PDF 1.4
mimetype - application/pdf
```

Figure 17. Output of PDF file after processing by libextractor

```
creation-date:  20080106161535-08'00'
producer:  OpenOffice.org 2.3
creator:  Impress
format:  PDF 1.4
mimetype:  application/pdf
```

Figure 18. Output of PDF file after processing by Libextract_plugin

V. ANALYSIS OF OPEN OFFICE AND OFFICE OPEN XML FILES

Chapter II provided an introductory look at the both Microsoft Office 2007 and Open Office file formats. This chapter builds upon the Office 2007 introduction and provides a comparative metadata analysis of the two document file types by examining timestamps, encryption, and thumbnails associated with each type.

A. EXAMINATION OF MICROSOFT OFFICE 2007 DOCUMENTS

1. Document.xml file

The `document.xml` file contains the main text and body of the document. Some of the important XML elements within this file include the following:

- `<w:document>` - root element; required to start defining document
- `<w:body>` - child element of `<w:document>`; text that comprises document is stored here
- `<w:p>` - paragraph within `<w:body>`, basic unit of document
- `<w:r>` - region of text with a common set of properties; paragraphs can be split into multiple runs, but runs must be contained within a paragraph, and can contain run properties, revision ids, and run content
- `<w:sdt>` - structured document tag; node is created when content control is added to a document; two sections within structured document node are properties and content sections.
- `<w:t>` - text element; document can have multiple `<w:t>` within a `<w:r>` [5, 6, 8]

2. Content Controls

Users can add various content controls (bounded and potentially labeled regions within a document that serve as containers for specific types of content) to their `.docx` files. Items such as dates, lists, and paragraphs of formatted text can be contained within individual content controls. Adding content controls to other content controls is also

possible. The controls provide the capability to create rich, structured blocks of content. They also build on the custom XML support introduced in Microsoft Office Word 2003 [15].

Additionally, content controls can be used to prompt users to provide data or to bind values to controls within the `document.xml` file from separate `.xml` files in the `.docx` package [8]. By inspection of Microsoft Word 2007, the types of content controls include the following:

- Rich text
- Plain text
- Picture content
- Combo box
- Drop down list
- Date picker
- Building block gallery
- Legacy controls such as Active X, text fields and check boxes

3. Identifiers

Office 2007 documents contain many types of identifiers. The paragraph revision id was examined in Chapter II. In this section, Relationship Ids and store item ids are discussed along with an explanation of how data and content are bound within a document.

The Relationship Id of two separate `.docx` documents created on the same computer were examined. The same relationship Id, `<Relationship Id="rId2"` was found in each. Uniqueness thus appears to mean "unique within the document." Rice confirms this deduction, by stating that the id can be any string as long as it is unique within the `.rels` file [45].

With the new file formats, users can add their own data and content by creating their own custom-defined xml and placing it in the file as another part [45]. The value of `<w:customXml>` is that the user can markup elements within the `document.xml` file,

which is useful for adding metadata/business semantics or for adding levels of granularity for search purposes [7]. The customized xml can be used with a tool or application that is capable of accessing and reading Office Open XML formats [45].

Data store items are used to distinguish the content pieces and to ensure the correct data is bound. The identifier is called the `store item id` and is attached to customXml by using a properties file [8].

The properties file defines the ID of the `customXml` part and the XML schema for the part. The property files are placed inside the `customXml` folder within the `.docx` package [8].

Each `customXml` part is related to its properties part through entries in `customXml/_rels` within the package. Assuming there are two custom parts - `item1.xml` and `item2.xml`, then the entries within `customXml/_rels` would be `item1.xml.rels` and `item2.xml.rels` [8].

The remaining piece that must be referenced is the link to the custom parts from the `document.xml` file. For this reference to hold, there will be a `<w:databinding>` element within the structured document properties [8].

`StoreItem` ids (introduced above) are assigned to pieces and referenced in the `document.xml` file. If the content is edited and saved again within Office 2007, new ids will be assigned. For instance, if the pieces are named `piece1.xml` and `piece2.xml`, they will be renamed to `item1.xml` and `item2.xml` once the document is edited and saved within Office 2007. All of the package references will be updated as well [8].

To an investigator, the various identifiers contained within an Office 2007 document provide a possible means of tracking user added content and data within a document. Of course, remembering that identifiers are only 32 bits long and that there is no way of guaranteeing that they are unique is important. There exists a $1:2^{32}$ chance that two specific documents will match by chance, and a much higher chance that at least two documents in a corpus will match – a direct result of the birthday paradox.

B. TIMESTAMPS

Time is frequently of critical importance in forensic investigations. Both Open Office and Office Open XML contain numerous internal timestamps indicating the time that documents were created or modified. Timestamps are present in the ZIP archive itself, (Figures 19, 20, 21) in the embedded XML files, and potentially in other embedded objects (for example, in the EXIF headers of embedded JPEGs). Unfortunately, not all of the timestamps are accurate. As demonstrated in Figure 21, NeoOffice (a port of OpenOffice.org to the Macintosh) sets the timestamps of the ODF file's ZIP archive to be the same as the system clock. The times are expressed in GMT, without a local time zone correction. Microsoft Word and Excel, on the other hand, set the timestamp on the ZIP archive to be January 1, 1980, the Epoch of the Microsoft FAT file system.

In addition to these ZIP directory timestamps, there are many different timestamps embedded within various XML sections, including:

- Word 2007, Excel and PowerPoint put the document's creation date in the tag of the `core.xml` file. The modified date is coded in the `dcterms:modified` element of the same file.
- PowerPoint 2007 additionally coded the document's creation date in the `a:fld` XML tag of each `slideLayout.xml` file.
- When change tracking was enabled in Word 2007, the XML file was annotated with multiple `w:ins` tags. Each tag included a `w:author` attribute with the editor's name, a `w:date` attribute with the date of the modification, and a `w:id` attribute with the ID number of the modification.
- A CNN.COM webpage pasted from a web browser into a Microsoft Word document and then saved as a `docx` file contained timestamps embedded in the CNN.com URLs.
- NeoOffice encoded the document's creation date in the `meta:creation-date` tag of the `meta.xml` section contained within blank `ODP`, `ODS` and `ODT` files.
- NeoOffice likewise embedded a `thumbnail.pdf` file inside blank `ODP`, `ODS` and `ODT` files. This `PDF` file included comments for a `CreationDate`.
- NeoOffice embedded the date in a `text:date` tag within the `styles.xml` file of a blank presentation.

```
Length      Date    Time    Name
 --------    ----    ----    ----
     1312  01-01-80 00:00   [Content_Types].xml
      590  01-01-80 00:00   _rels/.rels
      817  01-01-80 00:00   word/_rels/document.xml.rels
      985  01-01-80 00:00   word/document.xml
     6992  01-01-80 00:00   word/theme/theme1.xml
     1532  01-01-80 00:00   word/settings.xml
     1031  01-01-80 00:00   word/fontTable.xml
      260  01-01-80 00:00   word/webSettings.xml
      712  01-01-80 00:00   docProps/app.xml
      753  01-01-80 00:00   docProps/core.xml
    14840  01-01-80 00:00   word/styles.xml
 --------                   -------
    29824                   11 files
```

Figure 19. Contents of an empty Microsoft Word 2007 document (Windows environment). Note that none of the timestamps have been properly set.

```
Archive:  word1-save1.docx
  Length      Date    Time    Name
 --------    ----    ----    ----
     1364  01-01-80 00:00   [Content_Types].xml
      735  01-01-80 00:00   _rels/.rels
      817  01-01-80 00:00   word/_rels/document.xml.rels
     1062  01-01-80 00:00   word/document.xml
     7559  01-01-80 00:00   word/theme/theme1.xml
    12840  01-01-80 00:00   docProps/thumbnail.jpeg
     1963  01-01-80 00:00   word/settings.xml
     1521  01-01-80 00:00   word/fontTable.xml
      276  01-01-80 00:00   word/webSettings.xml
      710  01-01-80 00:00   docProps/core.xml
    15019  01-01-80 00:00   word/styles.xml
      734  01-01-80 00:00   docProps/app.xml
 --------                   -------
    44600                   12 files
```

Figure 20. Example docx file directory (Macintosh)

These timestamps might be significant in a forensic examination. For example, the timestamps potentially show when an ODF or Office Open XML file was edited with an ODF/Office Open XML-aware application. The timestamps might indicate multiple editing sessions. Alternatively, they might indicate tampering of a document. The timestamps might even be used in a file carver to determine which recovered pieces of a file match with other pieces.

```
Length     Date   Time    Name
 --------    ----   ----    ----
       39  02-21-08 18:26   mimetype
        0  02-21-08 18:26   Configurations2/statusbar/
        0  02-21-08 18:26   Configurations2/accelerator/current.xml
        0  02-21-08 18:26   Configurations2/floater/
        0  02-21-08 18:26   Configurations2/popupmenu/
        0  02-21-08 18:26   Configurations2/progressbar/
        0  02-21-08 18:26   Configurations2/menubar/
        0  02-21-08 18:26   Configurations2/toolbar/
        0  02-21-08 18:26   Configurations2/images/Bitmaps/
     2756  02-21-08 18:26   content.xml
     8678  02-21-08 18:26   styles.xml
     1004  02-21-08 18:26   meta.xml
      729  02-21-08 18:26   Thumbnails/thumbnail.png
     1043  02-21-08 18:26   Thumbnails/thumbnail.pdf
     7476  02-21-08 18:26   settings.xml
     1959  02-21-08 18:26   META-INF/manifest.xml
 --------                   -------
    23684                   16 files
```

Figure 21. ZIP directory for a NeoOffice (Mac) 2.2.2 ODT Word Processing file

C. ENCRYPTION

File encryption is another area that impacts an investigator's ability to extract metadata from files. During the research effort, only ODF and Microsoft Office 2007 documents were encrypted to determine the effect encryption had on the available metadata.

Open Office allows users to save a file with a password. The analysis of ODF files indicates that some sections are encrypted when a password is provided, but others are not.

To test the encryption, a NeoOffice file with a single line was created and saved with a password. Nine files were stored in the resulting archive. Of those files, the following were not encrypted:

- `META-INF/manifest.xml`
- `meta.xml`
- `mimetype`

The following files were encrypted:

- `Configurations2/accelerator/current.xml`
- `content.xml`
- `settings.xml`
- `styles.xml`
- `Thumbnails/thumbnail.pdf`
- `Thumbnails/thumbnail.png`

Revealing the contents of the encrypted file would require cracking the encryption algorithm or guessing the encryption password, but forensic investigators may find the information in the unencrypted sections useful nevertheless.

In the test document created with NeoOffice 2.2, the following XML tags might be potentially relevant to an investigation:

- `meta:generator`---The specific build of the specific application that created the document.
- `meta:creation-date`---The document's creation date in local time.
- `dc:language`---The document's primary language
- `meta:editing-cycles`---The number of times the document had been edited
- `meta:user-defined`---User-definable metadata (in this case these tags had the values `Info 1` through `Info 4`).
- `meta:document-statistics`---Including the number of tables, images, objects, page count, paragraph count, word count, and character count.

Microsoft Office 2007 allows users to password-protect documents as well. Four different Microsoft Word 2007 documents were created in a Windows environment. Each document was encrypted using a password. The documents created included one with just text; another with text and an embedded image file; and a third with text, a text content control object, and an image content control object. The fourth document contained text and manually entered metadata such as subject, keywords, and comments. The intent was to determine which files within the archive would be encrypted and which would not for a comparison to the ODF encryption results.

The examination revealed that encrypting the documents protected the archive. Neither Filzip [17] nor ZIP [49] recognized the encrypted document as a ZIP archive. However, the following files were extracted using 7Zip [42]:

- `EncryptionInfo`
- `EncryptedPackage`
- `WordDocument`
- `[6]Dataspaces/DataSpaceMap`
- `[6]Dataspaces/Version`
- `[6]Dataspaces/DataSpaceInfo/StrongEncryptionInfo`
- `[6]Dataspaces/DataSpaceInfo/TransformInfo/StrongEncryptionTransform/`
- `[6]Primary`

The contents of the above files did not reveal any meaningful information. However, the `EncryptionInfo` section contained the following readable text: `Microsoft Enhanced RSA` and `\AES Cryptographic Provider (Prototype)`.

The `Primary` file contained the following readable text:

`FF9A3F03-56EF-4613-BDD5-5A41C1D07246`

`Microsoft.Container.EncryptionTransform AES 128`.

Based on the analysis, encrypted Microsoft Office 2007 files appear to leak less information than ODF files, and therefore, would present move of a challenge for investigators needing to extract the metadata from these files.

D. THUMBNAILS

Embedded “thumbnail” images were found in the Microsoft Office files created by PowerPoint 2008, Excel 2008, and Word 2008 (Figure 20) on the Macintosh, as well as by NeoOffice. Thumbnails were also found in the `.pptx` created by PowerPoint 2007 on Windows (Figure 22). No thumbnail images created by Word 2007 (Figure 19) or Excel 2007 were encountered.

```
Archive:  pp1-save1-1.pptx
  Length      Date    Time    Name
 --------    ----    ----    ----
     3142  01-01-80 00:00   [Content_Types].xml
      738  01-01-80 00:00   _rels/.rels
      311  01-01-80 00:00   ppt/slides/_rels/slide1.xml.rels
      976  01-01-80 00:00   ppt/_rels/presentation.xml.rels
     3228  01-01-80 00:00   ppt/presentation.xml
     1072  01-01-80 00:00   ppt/slides/slide1.xml
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout7.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout8.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout10.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout11.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout9.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout1.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout2.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout3.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout4.xml.rels
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout5.xml.rels
     1991  01-01-80 00:00   ppt/slideMasters/_rels/slideMaster1.xml.rels
     3063  01-01-80 00:00   ppt/slideLayouts/slideLayout11.xml
     2839  01-01-80 00:00   ppt/slideLayouts/slideLayout10.xml
     4259  01-01-80 00:00   ppt/slideLayouts/slideLayout3.xml
     2784  01-01-80 00:00   ppt/slideLayouts/slideLayout2.xml
     4175  01-01-80 00:00   ppt/slideLayouts/slideLayout1.xml
    12065  01-01-80 00:00   ppt/slideMasters/slideMaster1.xml
     4530  01-01-80 00:00   ppt/slideLayouts/slideLayout4.xml
     7041  01-01-80 00:00   ppt/slideLayouts/slideLayout5.xml
     2051  01-01-80 00:00   ppt/slideLayouts/slideLayout6.xml
     1713  01-01-80 00:00   ppt/slideLayouts/slideLayout7.xml
     4620  01-01-80 00:00   ppt/slideLayouts/slideLayout8.xml
     4475  01-01-80 00:00   ppt/slideLayouts/slideLayout9.xml
      311  01-01-80 00:00   ppt/slideLayouts/_rels/slideLayout6.xml.rels
     7004  01-01-80 00:00   ppt/theme/theme1.xml
     2048  01-01-80 00:00   docProps/thumbnail.jpeg
      287  01-01-80 00:00   ppt/presProps.xml
      182  01-01-80 00:00   ppt/tableStyles.xml
      862  01-01-80 00:00   ppt/viewProps.xml
      675  01-01-80 00:00   docProps/core.xml
     1111  01-01-80 00:00   docProps/app.xml
 --------                   -------
    80663                   37 files
```

Figure 22. PowerPoint 2007 archive listing demonstrating presence of thumbnail image

The thumbnails were, without exception, images of the document's first page when rendered with the program that created the document file. The thumbnail images were examined for metadata themselves. The JPEG thumbnails only contained metadata for the image size and resolution (Figure 23), while the PDF thumbnails contained the thumbnail's creator, producer, and creation date (Figure 24).

The presence of a thumbnail image provides forensic investigators another avenue through which to extract metadata and information about the document. It is also potentially relevant for forensic investigators to know that Microsoft Office 2008 and Open Office generate thumbnail images while Microsoft Office 2007 does not.

```
Tag                 |Value
--------------------+------------------------------
x-Resolution        |72.00
y-Resolution        |72.00
Resolution Unit     |Inch
PixelXDimension     |256
PixelYDimension     |149
--------------------+------------------------------
```

Figure 23. EXIF fields from thumbnail.jpeg file contained within blank xlsx document created in Microsoft Word 2008 (Macintosh).

```
13 0 obj
<</Creator<FEFF0049006D00700072006500730073>
/Producer<FEFF004E0065006F004F006600660069006300650
0200032002E0032>
/CreationDate(D:20080311114631-07'00')>>
endobj
```

Figure 24. Header of a file embedded in NeoOffice thumbnail.pdf file. Creator is the UTF-16 coding of the word “Impress”, the Producer is the UTF-16 coding for NeoOffice 2.2, and the creation date is 2008-03-11 11:46:31 PDT.

VI. PRIOR AND RELATED WORK

This chapter discusses the extraction and uses of metadata in computer forensics and the use of metadata in searching. Specifically, the computer forensics tools EnCase and the Sleuthkit will be covered along with a discussion on file feature extraction, cross drive analysis and data mining. The chapter closes with a look at the use of metadata in programs such as FileHold, Google Search Appliance, and Oracle Data Integrator.

A. OTHER APPROACHES TO AUTOMATIC METADATA EXTRACTION

1. Metadata Extraction in EnCase

Encase is a popular forensics tool produced and maintained by Guidance Software. There are four classes of evidence files supported by EnCase Applications:

- EnCase Evidence Files (E01)
- Logical Evidence Files (LEF/L01)
- Raw Images
- Single files (including directories) [24]

The EnCase evidence files and single files are most interesting from a metadata standpoint. Evidence files contain the contents of an acquired device and provide the basis for future analysis. These files integrate investigative metadata, the device level-hash value and the content from the device. In contrast, the raw image files do not include the metadata or the hash values [24]. The single files (or directories) contain their existing metadata and can be added to the investigator's open case.

Guidance software added support for Microsoft Office 2007 documents (Microsoft Word, Excel, and Power Point) in EnCase Version 6.6. EnCase also has the ability to extract values in Excel worksheets created in Office Open XML file format [24]. In testing EnCase, we discovered that in addition to being able to extract text from Microsoft Office 2007 documents, we were also able to extract text from embedded files

within Microsoft Word documents. We tested a Word 2007 document that EnCase contained embedded files three layers deep, and a Microsoft Word 2003 document that contained embedded files four layers deep. In both cases, EnCase was able to extract the embedded text; where as, open source tools such as wvSummary and wvText were not.

In addition to Microsoft Office 2007 files, EnCase provides the functionality to view individual components of other compound file types such as registry files, OLE files, MS Outlook email files, Windows Thumbs.db, and Macintosh PAX files [24].

An important tool within EnCase is the *Case Processor*. The *Case Processor* allows users to run one or more *EnScript* (EnCase's internal scripting language) modules against an open case. From a metadata perspective, the most applicable modules include the following:

- EXIF Viewer
- Active Directory Information Parser
- File Finder
- File Report
- Find Protected Files
- HTML Carver [24]

The features within EnCase, like the modules identified above, aid in the collection of metadata such as name, filter, file type, file category, signature, description, file deleted, last access time, creation date/time, last written time stamp, and entry modified time stamp. Although not required, Jones, Bejtlich and Rose recommend calculating the MD5 hashes of the files prior to exporting the metadata to provide a means of verifying the integrity of the files. Once the hashes have been calculated, the user can select from a list of options regarding which metadata to export and where to export the results to [32].

2. Metadata Extraction in the Sleuthkit

The Sleuthkit (TSK) as introduced in Chapter III is an open source forensic tool for analyzing disk and file system images. TSK has no tools that process file metadata,

but TSK does have several tools that can process file system metadata: `istat`, `ils`, `ifind`, and `icat`. The `istat` tool provides details such as file size, temporal data, permission fields, and addresses of the allocated data units. The `ils` tool lists the details of several metadata structures [10].

Next, `ifind` is used when a data unit contains evidence with potential value and provides options for searching all metadata entries and for finding the metadata entry that a specific filename points to. Finally, `icat` allows the contents of any file to be viewed using the metadata address as opposed to the filename. The primary benefit of `icat` is that files no longer having a filename pointing to their metadata entry (unallocated files) can be retrieved and viewed [10].

B. USES OF METADATA IN COMPUTER FORENSICS

1. File Feature Extraction and Cross Drive Analysis

Garfinkel writes of two approaches for assisting investigators with simultaneously examining information across a collection of many data sources such as disk drives and solid-state storage devices. The two approaches are File Feature Extraction and Cross-drive analysis. Both of these approaches, when used together, allow investigators to quickly identify noteworthy drives and to correlate information between the different drives [20].

Cross-drive analysis relies on the identification and extraction of "pseudo-unique identifiers" like credit card numbers and email Message-IDs. The identifiers are called "features" and are used in the conduct of single-drive and multi-drive correlation [20].

The following list of properties is associated with good pseudo-unique identifiers:

- Long enough to reduced the likelihood of collisions
- Can be recognized by regular expressions without requiring parsing and semantic analysis
- Do not change over time
- Can be correlated with specific documents, people, or organizations [20]

Several programs for have been constructed for feature extraction. The programs scan the disk image, searching for the “pseudo unique features,” and then the results are placed in a file [20].

More specifically, the feature extractors utilize regular expressions and are compiled using `Flex`. The additional rules are written in C++. Instead of running the extractor programs directly on the raw images, Garfinkel developed an alternative process which involves making three passes with the `strings` program contained as part of the Free Software Foundation’s `binutils` distribution. The first pass with `strings` retrieves an 8 bit byte encoding; the second pass obtains a 16 bit big endian encoding, and the file pass extracts a 16 bit little endian encoding. The extractor programs are then run using the three files as input. This process significantly reduces the amount of data the extractors need to examine while increasing the amount of features that are extracted due to the fact that extractors written to identify 8 bit features can also recognize 8 bit features embedded within a 16 bit character set [20].

A “feature file” is created with the results of the extractor runs. Each line within the file contains the feature that was uncovered along with the surrounding context in which the feature was identified and the offset of the feature within the disk image. The context and the offset can be used by other tools that allow the user to focus on the area within the file system where the feature was discovered [20].

The programs include extractors for

- Email addresses
- Email message-ids
- Email Subjects
- Dates (extracts date and time stamps in varying formats)
- Cookies (identifies cookies from the “Set-Cookie: header” in web page cache files)
- Social Security Numbers in SSN <:> xxx-xx-xxxx and xxxxxxxxx formats
- Credit card numbers [20]

One of the primary benefits of using the feature extractors is a decrease in analysis time. Additionally, the features provide insight into ownership of the drives. For

instance, extracted features from disk images have been used to assist in the identification and reporting of information that should have been destroyed under the Fair and Accurate Credit Transactions Act [20].

Despite the advancements with file feature extraction, more work is needed. As an example, the existing cookie extractor can be expanded to retrieve cookies from "cookie jars." By using tools such as Sleuthkit to preprocess the disk images and extract all data files from the image followed by a scan using format-specific feature extractors, additional specificity can be obtained [20].

2. Uses of Metadata in Data Mining

Data mining is one area in which the use of metadata can improve efficiency and performance. Before driving to the specific roles of metadata within data mining, basic background information on data mining is presented.

Data mining is defined by Witten and Frank, to be the process of identifying patterns in data. The process must be at least semi if not fully automated [52].

Within data mining, machine learning is a new technology for mining knowledge from the data. More specifically, machine learning is about using algorithms to infer structure from the data and then validating the structure. One of the leading focal points of machine learning is developing new schemes to permit learning methods to take metadata into account in a meaningful way [52].

Metadata frequently involves relations between attributes, and three kinds of relations can be distinguished: semantic, functional, and casual. Casual relations exist when one attribute causes another. Using causal metadata as an example, under machine learning, the "system" should be able to deduce that A causes C if it is given that A causes B, and B causes C (where A, B, and C represent casual metadata) without having to state the fact explicitly [52].

Related to data mining is text mining. Data mining searches for patterns in data; where as text mining searches for patterns in text. Text is frequently unstructured and considered difficult to work with. Metadata extraction can be considered a general class

of text mining problems. One of the more difficult problems is authorship ascription, in which a document's author is unknown and must be "guessed" from the text. Using metadata can assist in building the correlation to an author [52].

The document owner ascription problem can be generalized to ownership of files on a hard drive. In his thesis work, Huynh explores the relationship of metadata found in files to ascribe the file to a potential owner of the file. His work utilizes metadata in the ascription process and seeks to determine which metadata (subject, creation date, modification date, etc) provides the best correlation between a file and a potential owner of the file [29].

In digital forensics, intelligence, law enforcement, and military organizations need a means to rapidly process and analyze captured information for evidence, and being able to use metadata within the data mining/ascription effort, improved this process. As Huynh's findings indicated metadata can be used to associate file to a specific user/owner on drives from systems containing multiple users [29].

C. USES OF METADATA IN SEARCHES

Lastly, there has been a lot of interest in metadata for search. The companies FileHold, Google, and Oracle utilize file metadata to improve the efficiency of their search engines. Improving the efficiency of searching is relevant to computer forensics because this technology may play a role in improving forensic applications.

1. FileHold

The company, FileHold has produced the File-Hold Search Engine, to save users time when searching through a library of files. The search engine indexes the `meta` tags associated with the document in addition to the content of the document. Data and content can be extracted from `PDF`, Microsoft Office, and `Zip` files among others [48].

The metadata search functionality within FileHold provides users with the capability of creating their own search queries using the *library administrator* defined

metadata fields such as document keywords, system users, file type, creation/modified dates, document approval status, document version, document number, and library location [48].

2. Google Desktop Search/Google Search Appliance

Google Desktop Search (GDS) is a local searching tool that allows users to index and search the contents of their computers. GDS offers the capability to index and search for files using metadata for file types such as multimedia files that do not generally contain content that can be readily indexed. In conjunction with GDS, Google also offers the Google Desktop Search Software Development Kit (SDK) for extending GDS with custom plug-ins. The extensibility of SDK provides support for new file types and expands the search facilities of existing applications [44].

Google also offers the Google Search Appliance (GSA). The GSA indexes metadata stored in documents and makes data available for retrieval at search time. Using metadata can improve the quality of a search. From the perspective of GSA, there are two types of metadata – in a primary document and not in a primary document (external metadata) [22].

GSA automatically indexes metadata in a primary document, and external metadata such as that found in a database table can be indexed as well. As with GDS, GSA offers APIs for developers to use to extend the reach of traditional indexing and searching [22].

3. Oracle Data Integrator

Oracle is another corporation that develops products that rely on metadata extracted from files. Metadata plays a important role in Oracle's information management initiatives such as MDM, customer data integration (CDI), product information management (PIM), and product data management (PDM). These initiatives center upon generating and maintaining a clean, accurate view of corporate reference data that is shared across operational and analytic systems [41].

One of Oracle's tools for managing metadata is the Oracle Data Integrator, which provides the mechanisms needed to "retrieve, enrich, extend, and leverage existing metadata for agile corporate enterprise architecture [41]."

The Oracle Data Integrator possesses a metadata repository, which can be installed on Oracle, Microsoft SQL Server, IBM DB2 UDB, IBM DB2/400, Informix, Sybase AS Anywhere, Sybase AS Enterprise, and Sybase ASIQ relational databases. The metadata is stored in database tables and can be used as a source by any reporting system [41].

The Oracle Data Integrator also includes a reverse engineering functionality that populates the repository with metadata from the information system. Reverse engineering extracts metadata from data storage as found in databases and XML files and stores the data into the repository. Nonstandard metadata can be retrieved from the databases or from proprietary repositories such as enterprise resource planning/customer relationship management (ERP/CRM) systems by customizing the reverse engineering process [41].

Finally, the Oracle Data Integrator can be a metadata-based code generator. The Oracle knowledge modules utilize the metadata stored in the repository. This metadata is used to create the appropriate code which is then run on existing systems. Because metadata is focal point of the integration and data integrity check processes, maintaining the processes is faster. As an example, the Oracle Data Integrator can automatically produce Adobe PDF reports using the repository contents to document the enterprise architecture and integration processes [41].

VII. CONCLUSION

A. FINDINGS

1. Automated Extraction

This thesis sought to determine whether or not existing metadata extraction tools could be combined for the automated processing of disk images. Through the analysis of different file formats and the examination of existing metadata extraction tools such as `exif`, `wv`, `libextractor`, and the specially created `docx_extractor`, I determined that by incorporating the plug-ins into the application `fiwalk`, metadata extraction tools could be combined for the automated processing of disk images.

2. Metadata Extraction Opportunities

This thesis also sought to determine what metadata extraction opportunities existed in a forensic context for various file formats. Through the research effort, document files (pre – 2007 Microsoft Office, Microsoft Office 2007, and Open Office) were found to contain metadata that would be interesting from a computer forensic perspective. While media files also contained metadata, the information available would be less useful for an investigator when compared to metadata available in document files. For instance, the metadata associated with an `.mp3` file (artist, title, year), which when compared to the metadata associated with a document file (file creator, revision number, modifier, description) is not as pertinent to a forensics investigation. However, some media files such as `jpegs` potentially contain metadata such as camera manufacturer and serial number, which an investigator would be interested in obtaining.

As previously discussed, Office 2007 documents provide a significant amount of information that can be extracted from a package beyond the standard author, creation/modification dates, that were obtained from prior versions of Office. The Office Open XML format presents the forensic investigator and forensic tools developer with a simpler environment in which to extract metadata. The presence of a thumbnail file or

other embedded image within the package provides investigators with additional metadata that can be extracted and analyzed. By examining `rsid` values within the XML, investigators can identify documents that were created during the same editing session but were later dispersed and modified separately.

3. Metadata Comparison

Because determining a clear cut metadata winner is not easily done, comparing similar attributes between Open Document Format (ODF – Open Office) and Office Open XML files presents a clearer picture of the metadata potentially available for forensic investigations. As discussed in Chapter V, three areas for direct comparison are timestamps, encryption, and thumbnails. During the research, timestamps were found in Open Office and Office Open XML documents. However, the time stamps were not always accurate.

Encryption was another area available for direct comparison between the document file types. During the research effort, only ODF and Microsoft Office 2007 documents were encrypted to determine the effect encryption had on the available metadata. Based on the analysis, encrypted Microsoft Office 2007 files appear to leak less information than ODF files, and therefore, would present more of a challenge for investigators needing to extract the metadata from these files.

The presence of thumbnails was an additional area for consideration when comparing the metadata of the document file formats. ODF files contain both `.jpeg` and `.pdf` thumbnails while the Macintosh version of Microsoft Office 2008 contains `.jpeg` thumbnail images in the document archive. In contrast, the Windows version of Microsoft Office 2007, only contributes a thumbnail image in the PowerPoint 2007 document archives.

4. Deficiencies in Metadata Extraction Tools

Another objective of this thesis was to ascertain whether or not existing open source metadata extraction tools generated accurate results. During the research effort, several shortcomings with the metadata extraction tools were encountered. Understanding

that these deficiencies exist is important for forensic investigators because some results may require additional analysis or research. A discussion of the deficiencies is discussed below.

WvSummary proved to be useful in extracting most metadata from pre – 2007 Microsoft Office documents. However, some inconsistencies and inaccuracies were encountered. For instance, the document analyzed in Figure 9 was comprised of three pages, but wvSummary reported that the document contained one page. This inaccuracy was encountered in other test files as well. However, the number of worksheets and slides, including hidden slides, was accurately reported by wvSummary.

Also, within the document analyzed in Figure 9, comments were manually entered into the file, but comments were not one of the extracted pieces of metadata retrieved by wvSummary. WvSummary did capture customized metadata items such as Keywords, Edited, Received from and Checked by. The metadata added to the Excel and PowerPoint files was also captured by wvSummary.

Both pre and post Microsoft Word 2007 applications allow other Word documents to be embedded within a Word Document using the Insert/Object... menu command. To test the effectiveness of the metadata extraction tools, a Word document was embedded within a .doc and a .docx document. Figure 25 provides the ZIP archive of a .docx document with another Word file embedded within it. In the case of the .doc file, wvSummary and its sister tool wvText failed to identify the embedded file. Similarly, docx_extractor failed to identify the embedded document within the .docx archive. This test highlighted a bug within docx_extractor that needs to be fixed. Of note, contrary to the results observed from the open source wvSummary/wvText, Guidance Software's EnCase found the embedded document in both formats.

Issues were also encountered when using libextractor. In the initial trials, limited metadata was retrieved. The test files included .gif, .jpg, .pdf, and .html files. The metadata obtained included filename (using –f option), file size, and mime type. Libextractor claims to provide an extensive list of keywords that can be matched within the metadata, but initial trials did not retrieve metadata such as title or comments.

Additional libraries and plug-ins are available for use with `libextractor`, but locating the missing components and deriving the desired configuration and results were not achieved.

```
Length   Name
-------  ----
   1527  [Content_Types].xml
    735  _rels/.rels
   1107  word/_rels/document.xml.rels
   4780  word/document.xml
   6613  word/media/image1.png
   7559  word/theme/theme1.xml
  39832  docProps/thumbnail.jpeg
  25316  word/embeddings/Microsoft_Word_Document1.docx
   2036  word/settings.xml
    276  word/webSettings.xml
    734  docProps/app.xml
    726  docProps/core.xml
  15019  word/styles.xml
   1521  word/fontTable.xml
-------  -------
 107781  14 files
```

Figure 25. ZIP archive of .docx document with embedded .docx document

Unfortunately, the `Exif` program utilized in `jpeg_extract` also demonstrated shortcomings. `Exif` processed digital images taken with a Canon EOS Rebel camera without any issues, but digital pictures taken with the Olympus FE-280 and Olympus Stylus 400 cameras were not recognized as being EXIF format.

B. FUTURE WORK

Given more time, I would incorporate additional functionality into `fiwalk`. For instance, adding automated feature extraction would provide a positive supplement to the metadata extraction capabilities provided by the plug-ins. Additionally, developing more metadata extraction plug-ins and modifying `fiwalk` to process all content with the plug-ins as opposed to just named files would increase the range of `fiwalk`. Currently, a plug-in for extracting metadata from Open Office documents has been written, but the plug-in needs additional work to ensure that metadata buried deep in the XML trees is recovered. Once the plug-in is completed and tested, it should be incorporated within `fiwalk`.

Under the current framework, metadata is not recursively extracted from every container encountered. By adding this feature, metadata from files such as ZIP and tar archives as well as metadata from embedded `.jpeg` and document files inside `.docx` archives would be available for forensic investigators to perform more in-depth analysis. Another needed capability is mapping extracted sectors and automatically carving the remainders.

The number of files on a disk image processed by `fiwalk` could theoretically number into the millions. Many of these files belong to the operating system and common user applications. As a consequence, an investigator most likely will not be concerned with analyzing these files. The National Software Reference Library (NSRL) sponsored by the U.S. Department of Justice's National Institute of Justice and the National Institute for Standards and Technology provides a repository of known software, file profiles, and file signatures for use by law enforcement and other organizations in computer forensics investigations [38]. By modifying `fiwalk` to screen files from a disk image against the repository, known files can be eliminated from review, potentially saving an investigator many hours of analysis time.

Although `fiwalk` provides the option to produce the output in multiple formats such as ARFF and soon will support XML, being able to process the output as SQL would enable the user to automatically populate a database with the metadata extracted from the files on the disk image for further analysis and faster data retrieval.

One limitation of working with metadata is that all metadata is susceptible to tampering. For instance, Office 2007 provides a means for programmers to search and remove metadata and content from a `.docx` document. The effect on an investigator is that information that may have been previously extracted from an Office document left behind by a naive user may no longer be present. As another example, some file timestamps can be manipulated by individuals trying to distort or modify the timeline history of a file. A useful tool or plug-in for investigators would be one that detects and reports instances of tampering of metadata and if possible recovers the original metadata.

Finally, based on the work initiated by Huynh in the area of file owner ascription, being able to automatically feed the output of `fiwalk` into a another plug-in, a data mining tool such as `Weka`, or an automated reporting tool would significantly increase the rate at which digital forensic investigators, law enforcement, intelligence and military organizations could process the data and derive meaningful results.

LIST OF REFERENCES

[1] Adobe Systems Incorporated. (2001). *Portable document format reference manual version 1.4*. San Jose, CA: Adobe Systems Incorporated.

[2] Apple Support. (10/31/2007). *Core Audio Glossary*. Retrieved 1/18/2008, from http://developer.apple.com/documentation/MusicAudio/Reference/CoreAudioGlossary/Glossary/chapter_998_section_1.html.

[3] Apple Support. (2003). *MPEG4. The new standard for multimedia on the Internet, powered by QuickTime*. Apple Incorporated.

[4] Apple Support. (2007). *Creating content for iPod + iTunes*. Retrieved 1/19/2008, from http://images.apple.com/support/itunes_u/docs/iTunes_U_Creating_Content.pdf.

[5] E. Ashton, & J. Hietianiemi. (2007). *The CPAN frequently asked questions.* Retrieved 4/3/2008, from http://www.cpan.org/misc/cpan-faq.html.

[6] P. Aven. (2007). *Running (a.k.a. <w:r>-ing) with word. Part 4 in a series on MarkLogic server and office 2007*. Retrieved 2/24/2008, from http://xqzone.marklogic.com/columns/smallchanges/2007-12-18.xqy.

[7] P. Aven. (2008). *Enriching word documents with <w:CustomXML>. Part 5 in a series on MarkLogic server and office 2007.* Retrieved 2/21/2008, from http://xqzone.marklogic.com/columns/smallchanges/2008-01-08.xqy.

[8] P. Aven. (2008). *A final word. Part 6 in a series on MarkLogic server and office 2007*. Retrieved 2/21/2008, from http://xqzone.marklogic.com/columns/smallchanges/.

[9] B. D. Carrier. (2003). *The sleuthkit informer, issue 1*. Retrieved 1/29/2008, from http://www.sleuthkit.org/informer/sleuthkit-informer-1.txt.

[10] B. D. Carrier. (2005). *File System Analysis.* Addison-Wesley.

[11] B. D. Carrier. (2003). *The sleuthkit informer, issue 22*. Retrieved 3/29/2008, from http://www.sleuthkit.org/informer/sleuthkit-informer-22.txt.

[12] B. D. Carrier. (2006). *Risks of live digital forensic analysis*. Communications of the ACM, 49(2), 56-61.

[13] CompuServe Incorporated. (1990). *GIF graphics interchange format, version 89a*. Retrieved 1/20/2008, from http://www.w3.org/Graphics/GIF/spec-gif89a.txt.

[14] ECMA. *Office Open XML File Formats Standard -Final Draft* - 9th of October 2006. Retrieved 3/7/2008, from http://www.ecma-international.org/news/TC45_current_work/TC45-2006-50_final_draft.htm.

[15] E. Ehrli, L. Wollin, & B. Jones. (2008). *Building word 2007 document templates using content controls.* Retrieved 5/10/2008, from http://msdn.microsoft.com/en-us/library/bb264571.aspx?ref=carstuning.biz.

[16] D. Farmer, & W. Venema. (2005). *Forensic Discovery*. Addison-Wesley.

[17] *Filzip*. Retrieved 5/24/2008, from http://filzip.com.

[18] G. A. Francia, & K. Clinton. (2005). *Computer forensics laboratory and tools*. J.Comput.Small Coll., 20(6), 143-150.

[19] S. L. Garfinkel. (2006). *AFF: A new format for storing hard drive images*. Communications of the ACM, 49(2), 85-87.

[20] S. L. Garfinkel. (2006). *Forensic feature extraction and cross-drive analysis*. Digital Investigation, 3(Supplement 1), 71-81.

[21] S. L. Garfinkel, D. Malan, K. Dubec, C. Stevens, & C. Pham. (2006). *Disk imaging with the advanced forensic format, library and tools*. Paper presented at the Second Annual IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL USA. Retrieved 1/18/2008, from http://www.simson.net/cv/pubs.php.

[22] *Google search appliance external metadata indexing guide*. Retrieved 1/19/2008, from http://code.google.com/enterprise/documentation/metadata.html.

[23] C. Grothoff. (2005). *Reading file metadata with extract and libextractor*. Retrieved 1/18/2008, from http://www.linuxjournal.com/article/7552.

[24] Guidance Software Incorporated. (2007). *EnCase forensic version 6.8 user manual*. Unpublished manuscript.

[25] S. Harris. (2005). *All-in-one CISSP Exam Guide, third edition*. Emeryville, CA:McGraw Hill.

[26] *Hidden data in JPEG files*. Retrieved 1/18/2008, from http://netzreport.googlepages.com/hidden_data_in_jpeg_files.html.

[27] *How to read compound document properties directly with VC++*. Retrieved 1/19/2008, from http://support.microsoft.com/kb/q186898.

[28] K. Husain. (2007). *Extract and parse ODF files with python*. Retrieved 1/18/2008, from http://www.linuxjournal.com/article/9347.

[29] D. Huynh. (2008). *Exploring and validating data mining algorithms for use in data ascription.* Unpublished MS Computer Science, Naval Post Graduate School.

[30] *ID3v2Easy - ID3.org*. Retrieved 1/25/2008, from http://www.id3.org/ID3v2Easy.

[31] JEITA. (2002). JEITA CP-3451. *Exchangeable Image File Format for Digital Still Cameras: EXIF version 2.2*. Japan Electronics and Information Technology Industries Association: Adobe Systems Incorporated.

[32] K. J. Jones, R. Bejtlich, & C. Rose. (2006). *Real Digital Forensics. Computer Security and Incident Response*. Upper Saddle River, NJ: Addison-Wesley.

[33] D. Lachowicz, & C. McNamara. *wvWare, library for converting word documents.* Retrieved 1/19/2008, from http://wvware.sourceforge.net/.

[34] Lutz M. *SourceForge.net: EXIF tag parsing library.* Retrieved 4/20/2008, from http://sourceforge.net/projects/libexif.

[35] N. A. Mikus. (2005). *An analysis of disc carving techniques [electronic resource]*. Monterey, Calif; Springfield, Va: Naval Postgraduate School; Available from National Technical Information Service.

[36] *MPEG4IP - open streaming video and audio*. Retrieved 1/21/2008, from http://mpeg4ip.sourceforge.net/documentation/index.php.

[37] S. Nagel. (2004). *Embedded information in electronic documents*. Retrieved 1/21/2008, from http://www.abanet.org/lpm/lpt/articles/ftr07044.html.

[38] National Institute of Standards and Technology. (2008). National software reference library. Retrieved 4/5/2008, from http://www.nsrl.nist.gov/.

[39] *OLE concepts and requirements overview*. Retrieved 1/19/2008, from http://support.microsoft.com/kb/86008/en-us.

[40] D. ONeil. *Home - ID3.org*. Retrieved 1/22/2008, from http://www.id3.org/.

[41] *An oracle data integrator technical briefing: Managing integration metadata*. (2006). Retrieved 1/19/2008, from http://209.85.173.104/search?q=cache:mgYvbbThP7YJ:www.eticasoft.com/wp/odi/oracledi_metadata.pdf+managing+metadata+with+oracle+data+integrator&hl=en&ct=clnk&cd=3&gl=us.

[42] I. Pavlov. *7-zip.* Retrieved 5/24/2008, from http://www.7-zip.org/.

[43] *Public private peer review and Microsoft Word* « David W. Boles' urban semiotic ™. Retrieved 1/25/2008, from http://urbansemiotic.com/2006/04/07/public-private-peer-review-and-microsoft-word/.

[44] L. Reeve. *Dr. Dobb's Improving search precision using Google desktop search 1.0* | November 1, 2005. Retrieved 1/30/2008, from http://www.ddj.com/database/184406319.

[45] F. Rice. (2006). *Introducing the office (2007) open XML file formats*. Retrieved 1/27/2008, from http://msdn2.microsoft.com/en-us/library/aa338205.aspx.

[46] *Reading ID3 tags with perl's MP3::Tag module*. Retrieved 1/23/2008, from http://articles.techrepublic.com.com/5100-22-5293815.html.

[47] *The risks of metadata and hidden information*. (2007). Retrieved 1/21/2008, from http://209.85.173.104/search?q=cache:aYiqoYmIAbYJ:www.oracle.com/technologies/embedded/docs/OutsideIn-MetadataRisks.pdf+risks+of+metadata+and+hidden+information&hl=en&ct=clnk&cd=1&gl=us.

[48] *Search and retrieval functions: Document management software*. Retrieved 1/19/2008, from http://www.filehold.com/product/feature-search.htm.

[49] C. Spieler. *UnZip 5.52.* Retrieved 5/24/2008, from http://www.WinZip.com.

[50] J. E. Towle, C. T. Clotfelter. (2007). *TwiddleNet [electronic resource]: Metadata tagging and data dissemination in mobile device networks*. Monterey, Calif: Naval Postgraduate School.

[51] M. Ward. *BBC NEWS | technology | tools reveal secret life of documents*. Retrieved 1/25/2008, from http://news.bbc.co.uk/2/hi/technology/3037760.stm.

[52] I. H. Witten, & E. Frank. (2005). *Data mining practical machine learning tools and techniques* Morgan Kaufmann.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California