JOURNAL OF FORENSIC SCIENCES

J Forensic Sci, 2014
doi: 10.1111/1556-4029.12528
Available online at: onlinelibrary.wiley.com

TECHNICAL NOTE

DIGITAL & MULTIMEDIA SCIENCES

Simson L. Garfinkel,[1] *Ph.D.*

The Prevalence of Encoded Digital Trace Evidence in the Nonfile Space of Computer Media*,†,‡

ABSTRACT: Forensically significant digital trace evidence that is frequently present in sectors of digital media not associated with allocated or deleted files. Modern digital forensic tools generally do not decompress such data unless a specific file with a recognized file type is first identified, potentially resulting in missed evidence. Email addresses are encoded differently for different file formats. As a result, trace evidence can be categorized as Plain in File (PF), Encoded in File (EF), Plain Not in File (PNF), or Encoded Not in File (ENF). The tool *bulk_extractor* finds all of these formats, but other forensic tools do not. A study of 961 storage devices purchased on the secondary market and shows that 474 contained encoded email addresses that were not in files (ENF). Different encoding formats are the result of different application programs that processed different kinds of digital trace evidence. Specific encoding formats explored include BASE64, GZIP, PDF, HIBER, and ZIP.

KEYWORDS: forensic science, digital forensics, optimistic decompression, *bulk_extractor*, real data corpus, encoded nonfile, Microsoft Xpress, BASE64, GZIP, PDF, ZIP

This study demonstrates that forensically significant digital trace evidence that is *compressed* or otherwise *encoded* is frequently present in sectors of digital media that are not associated with allocated or deleted files. This finding is important, because modern digital forensic tools generally do not decompress or otherwise decode bytes in unallocated sectors unless a specific file with a recognized file type is first identified. As a result, today's digital forensic tools potentially miss important evidence.

It has long been established (1,2) that a variety of information can be present in Nonfile (NF) space on digital media. Such information includes:

- Files that have been deleted and have had their file system metadata overwritten, such that they can no longer be readily identified. (Such files can sometimes be recovered through file carving).
- Files that were previously written to the disk and have since been partially overwritten, so that the entire file cannot be recovered, not even with carving.
- Remnants of files that have been relocated as a result of file system defragmentation operations, such that some sectors are still recoverable.
- Remnants of files from previous file systems, after which the drive was subsequently reformatted or damaged.

[1]Department of Computer Science, Naval Postgraduate School, 1186 North Utah Street, Arlington, VA 22201-4758.

*Presented in part at the 65th Annual Meeting of the American Association of Forensic Sciences, February 18–23, 2013, in Washington, DC.
†Funded by the U.S. Department of Defense.
‡The views presented in this article are those of the author and do not necessarily represent the views of the Department of Defense or its components.

Received 2 April 2013; and in revised form 29 June 2013; accepted 13 July 2013.

Previous research (3,4) also established at least five potential sources of compressed or otherwise encoded data on digital media:

- Many web browsers download data from web servers as compressed streams and store these streams directly in the web cache.
- NTFS file compression may result in disk sectors that contain compressed data.
- Windows hibernation files, compressed with Microsoft's XPress algorithm (5), are defragmented by Windows background tasks.
- Files are frequently bundled into archive formats that employ compression (e.g., ZIP, RAR, and *.tar.gz*).
- The *.docx* and *.pptx* file formats used by Microsoft Office store content as compressed XML files in ZIP archives (6).

In all cases, when these files are relocated during the course of defragmentation, or when they are deleted and partially overwritten, compressed data can be left in unallocated sectors.

The phrases *optimistic decompression* (3) and *optimistic decoding* refer to a data analysis approach in which a sequence of bytes is examined to see whether it can be decompressed or otherwise decoded. If so, the bytes are decoded and processed. This approach is "optimistic" because the software proceeds with the assumption that the decompression or decoding will be successful, and the results are interpreted, even if there is corruption or truncation that might prevent the recovery of the entire original data stream.

This study gauges the overall usefulness of optimistic methods by examining the results of their application to a corpus of more than a thousand images from hard drives, USB storage devices, and flash cards (referred to here as "drives" or "drive images").

Clearly, the value of optimistic techniques depends on the subject media under examination. A drive that consists solely of blank sectors will not benefit from the technique, but a drive that

has been heavily used may contain important trace evidence that can be revealed through no other approach. Optimistic methods are generally unknown to today's digital forensics practitioners and unimplemented by today's digital forensics tools. The purpose of this article is to present the techniques and experimentally determine their usefulness for recovering digital trace evidence on a variety of media.

Materials and Methods

The term "*digital trace evidence*" is frequently used to describe digital evidence that might have high probative value in a forensic investigation. Examples of digital trace evidence include email addresses, credit card numbers, and Internet search terms.

Although digital trace evidence may be insufficient to definitively confirm or deny a hypothesis of an activity, the evidence can be used for corroboration or for the production of new leads. Thus, digital trace evidence is most useful during the investigation phase of a new case.

If optimistic techniques are generally useful for processing digital trace evidence, then there should be digital trace evidence that can be recovered no other way. This article proves that hypothesis by showing that many used disk drives contain email addresses in compressed data streams that are not contained within recoverable files. These email addresses can only be recovered through optimistic means.

The Conventional Forensic Pipeline

Modern computer forensic tools employ more-or-less the same approach to process digital media. We call this approach the *forensic pipeline*. The pipeline can be applied directly to subject media, ideally connected to the examiner's computer with a write-blocker to prevent accidental media compromise, or it can be applied to a sector-for-sector copy (a *disk image*) of the original media.

The forensic pipeline starts with the tool attempting to identify disk partition and file system structures, collectively referred to as filesystem metadata. Once identified, the pipeline enumerates every directory and file on the disk image, each directory is scanned, and each file is identified. For each file, the file type is determined, text is extracted and optionally indexed, pictures and videos are processed into thumbnails, and other format-specific steps are executed. Because of varying engagement rules, most of today's tools can be programmed to process allocated files, or both allocated and deleted files.

This top-down processing of computer media mirrors the way that a layperson would most likely analyze the contents of a drive. The process is easy to teach, easy to practice, and easy to explain in court.

In many cases, additional steps are employed to recover evidence from sectors that are unallocated and cannot be mapped to deleted files or directories. This article refers to such sectors as the NF space. This space is typically processed with regular expressions to scan for email addresses, credit card numbers, and other kinds of recognizable text, and with file carvers such as Adroit (7) and PhotoRec (8) to recover digital images, movies, and other kinds of media.

Conventional Extraction of Digital Trace Evidence

This section demonstrates why the simplistic strategy described in the previous section does not work for many types of files currently in use on digital computers.

Figure 1 shows Microsoft Word for Mac 2011 running on an Apple Macintosh computer with an example document created for this article. The document consists of the single sentence, "One two three user@comapny.com four five six" followed by two blank lines. This file was saved three times: as a Microsoft Word Binary File Format (9) (*word1.doc*); as a Office Open XML (10) formatted files (*word1.docx*); and as an Adobe Portable Document Format (11) file (*word1.pdf*) file generated directly from Microsoft Word (Fig. 2).

The 16-byte sequence user@company.com appears twice in the file *word1.doc*: first in a *mailto:* link at offset 2595 and a second time at offset 2614 (Fig. 3). Both of these are evident in the hexdump starting at location a00 shown in Fig. 3.

The file *word1.docx* does not obviously include the email address. Microsoft's ".docx" files are actually ZIP archives, and the *word1.docx* archive contains 13 embedded files, including a thumbnail of the document's first page, a table of contents, a table fonts, and other associated metadata (Fig. 4). The archive also includes a compressed XML file called *word/document.xml*. This compressed XML file at inside *word1.docx* at decimal offset 2451 and extends to offset 3225, for a total of 774 compressed bytes (Fig. 5). If the embedded file is extracted from the ZIP archive and decompressed, the result is 1990-byte XML file with a email address at offset 1500 (Fig. 6).

Finally, the file *word1.pdf* is a PDF version 1.3 file that contains three compressed binary streams occupying byte ranges 79–390, 713–3226, and 4278–21696. The first of these streams contains the email address in question, but it is encoded. An ASCII representation of the first 407 bytes of the file showing the first stream, still compressed, appears in Fig. 7. The stream is a ZLIB (12) compressed binary object. Decompressing the object reveals the ASCII stream shown in Fig. 8, which is a series of Adobe PDF commands. The text can be recovered by combining the letters between the parentheses, producing the string "One two three user@company.com four five six."

All three files contain the email address user@company.com, but each one encodes the trace evidence differently. Programs that understand these file types (e.g., Oracle's Outside In (13)) can extract the email address from all three files, provided that the files are intact. However, an examiner searching a disk with regular expressions or visually scanning the disk with a hex editor will discover only the email address from the file *word1.doc*, as the sectors associated with the file *word1.docx* contain bytes corresponding to compressed XML, while the sectors for *word1.pdf* contains a compressed sequence of PDF commands.

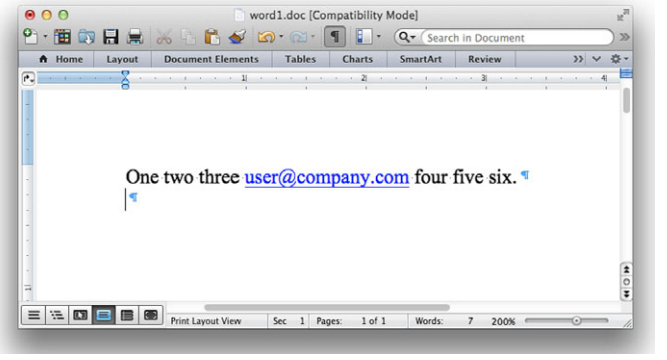

FIG. 1—*A Microsoft Word file containing a single sentence followed by a blank line.*

```
 filename     size                        SHA1
============================================================
word1.doc    22016  8174c1df7dc2f67f2b50d65f2d050dd8529b5d2f
word1.docx   24768  1b3f1ffe2b232587b1dba8a4407f0ef8ec171feb
word1.pdf    23001  47b0eb8475291a9236a123a6d8cc2ee108abc928
```

FIG. 2—*Three files resulting from saving the Microsoft Word file shown in 1 as a ".doc" file, a ".docx", and a ".pdf.*

```
00000a00: 4f6e 6520 7477 6f20 7468 7265 6520 1320  One two three .
00000a10: 4859 5045 524c 494e 4b20 226d 6169 6c74  HYPERLINK ‘‘mailt
00000a20: 6f3a 7573 6572 4063 6f6d 7061 6e79 2e63  o:user@company.c
00000a30: 6f6d 2220 1475 7365 7240 636f 6d70 616e  om’’ .user@compan
00000a40: 792e 636f 6d15 2066 6f75 7220 6669 7665  y.com. four five
00000a50: 2073 6978 2e0d 0d00 0000 0000 0000 0000   six.............
00000a60: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000a70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

FIG. 3—*Hexdump of a portion of word1.doc showing the embedded email addresses.*

```
Archive:  word1.docx
  Length      Date    Time    Name
 --------    ----    ----    ----
     1474  01-01-80 00:00   [Content_Types].xml
      735  01-01-80 00:00   _rels/.rels
     1118  01-01-80 00:00   word/_rels/document.xml.rels
     1990  01-01-80 00:00   word/document.xml
     7643  01-01-80 00:00   word/theme/theme1.xml
    11256  01-01-80 00:00   docProps/thumbnail.jpeg
     2432  01-01-80 00:00   word/settings.xml
      431  01-01-80 00:00   word/webSettings.xml
    16049  01-01-80 00:00   word/stylesWithEffects.xml
      759  01-01-80 00:00   docProps/core.xml
    15183  01-01-80 00:00   word/styles.xml
     2023  01-01-80 00:00   word/fontTable.xml
      740  01-01-80 00:00   docProps/app.xml
 --------                   -------
    61833                   13 files
```

FIG. 4—*The file word1.docx is actually a ZIP archive; this listing shows the archive's component files.*

Augmenting Extraction with Optimistic Decompression

Digital forensics tools that perform optimistic decompression operate by searching for byte patterns indicative of compressed data. When these byte patterns are identified, the tool attempts to decompress the data. Any resulting data are then analyzed.

The preceding section presents two cases in which it is possible to recover forensic trace information through the use of optimistic decompression. In the case of *word1.docx*, a useful strategy is to search for ZIP local file headers and attempt to decompress the compressed file data. In case of *word1.pdf*, a useful strategy is to search for the six-character sequence "stream" followed by a newline or a carriage return/newline pair, a high entropy region, and finally, the nine-character sequence "endstream". (Please see (14) for a discussion of the ZIP archive file format.) The high entropy region is then provided to zlib (12) for attempted decompression. Sections of the PDF that are successfully decompressed are then processed by a text extraction framework, which builds strings by combining the characters between parentheses. In this way, it is possible to extract from encoded sections of a file even if the entire file is not present or otherwise recoverable.

A Taxonomy of Digital Trace Evidence

Here, we present a classification scheme for describing how trace evidence may be present on digital media. Because trace evidence may be in a file or not in a file, and it may be in plain text or encoded, trace evidence may thus exist on the subject media in one of four conditions shown in Table 1.

In many cases, the same trace evidence is present in multiple locations on target media. For example, an email address might be downloaded as a compressed file but then decompressed in memory, and the memory might be written to the system swap partition. In such a case, there would be at least two copies of

```
00000990: 0300 504b 0304 1400 0600 0800 0000 2100  ..PK..........!.
000009a0: ea76 7d78 d702 0000 c607 0000 1100 0000  .v}x............
000009b0: 776f 7264 2f64 6f63 756d 656e 742e 786d  word/document.xm
000009c0: 6ca4 55db 729b 3010 7def 4cff 81d1 7b0c  l.U.r.0.}.L...{.
000009d0: 7673 7198 e034 b7a6 79e8 3453 b7cf 1d19  vsq..4..y.4S....
000009e0: 0468 8cb4 1a49 98ba 5fdf 95b8 d889 ddd6  .h...I.._.......
000009f0: 495e 0c98 b367 cf9e 5d2d 1797 bf44 15ac  I^...g..]-...D..
00000a00: 9836 1c64 42c6 a388 044c a690 7159 24e4  .6.dB....L..qY$.
00000a10: c7f7 4f47 5312 184b 6546 2b90 2c21 6b66  ..OGS..KeF+.,!kf
```

FIG. 5—*The ZIP local file header and compressed data for the word/document.xml component inside the file word1.docx. Each ZIP component begins with a component header consisting of the hexadecimal sequence 50 4B 03 04 ("PK.." above) and ending with a variable length name field ("word/document.xml") and an optional "extra" field (not present in this case). ZLIB-compressed data begins at offset 9c1 in the above example.*

```
w:t></w:r><w:hyperlink r:id=''rId5'' w:history=''1''><w:r w:rsidRPr=
''004B377A''><w:rPr><w:rStyle w:val=''Hyperlink''/></w:rPr><w:t>user
@company.com</w:t></w:r></w:hyperlink><w:r><w:t
```

FIG. 6—*ASCII dump of a portion of the file word/document.xml after being decompressed and extracted from the file word1.docx; the string user@company.com occurs at offset 5dc. (Line breaks have been added for legibility but do not occur in the source document.)*

```
%PDF-1.3
%\304\345\362\345\353\247\363\240\320\304\306
4 0 obj
<< /Length 5 0 R /Filter /FlateDecode >>
stream
x^A\225\222\313n\2030^PE\367\376\212\2734\213:\266^C^FvU\252n
\272\251''Y\352\242\352\242BAi^U\240\201\246\217\277\257\237
\224''\224\250^B\311^^{4>\367\316^\\261\305^QB\332?+\344\252
@\277\303^CZ\254n^F\201j^@w\337P\231<\316d\352c\273)9r^\260R
\241j\260\321$\363\231a\321^MVZ^K\306!\240k<\202\336'\2702^U
@\333]bK\201\342=l>\343U^W\216^H\335\3671+\256H\360^D}\207[m
\240\355[^B^KTBqV\346\251\372e#^[\215Kl\247!^R\304\327\372k
\313&@\253\300\3305      q\324o\317\341\244\375f\223\313^Y^\
_\202\223Y\311\224JE\200#\316\270\363p\324\310\326\257^\364
\274^CR\311\377\2263}\250\235\324^L\264;\365^X\203\227 \352
\303^[@(FK\342\325^W\233v‘^N\263\226)[\313''j\363C\361W^[\231
^M\305T\333\330P\241\314\320\244\306\241\234\311^B^M\234gf2}|
^H1\261\261I;‘\222\357\342=j?\243K^M03\253\312\314^V^\260\376
^\\336\366^G\212q\250^D
endstream
endobj
```

FIG. 7—*ASCII representation of the first 407 characters of the file word1.pdf showing the first compressed stream at byte offsets 79–390. (Line breaks have been added for legibility but do not occur in the source document.)*

```
q Q q 12 12 588 768 re W n /Cs1 cs 0 0 0 sc q 0.24 0 0 0.24 90 708.96
cm BT 50 0 0 50 0 0 Tm /TT1.0 1 Tf [ (O) -0.2 (ne) 0.2 ( t) 0.2 (w)
  -0.2 (o t) 0.2 (hre) 0.2 (e) 0.2 ( ) ] TJ ET Q 0 0 1 sc q 0.24 0 0
0.24 160.9746 708.96 cm BT 50 0 0 50 0 0 Tm /TT1.0 1 Tf [ (us) -0.2
  (e) 0.2 (r@) 0.1 (c) 0.2 (om) 0.2 (pa) 0.2 (ny.c) 0.2 (om) ] TJ ET Q
0 0 0 sc q 0.24 0 0 0.24 259.6641 708.96 cm BT 50 0 0 50 0 0 Tm /TT1.0
1 Tf ( ) Tj ET Q q 0.24 0 0 0.24 262.6641 708.96 cm BT 50 0 0 50 0 0
Tm /TT1.0 1 Tf [ (f) -0.5 (our f) -0.5 (i) 0.2 (ve) 0.2 ( s) -0.2 (i)
  0.2 (x.) ] TJ ET Q q 0.24 0 0 0.24 324.3281 708.96 cm BT 50 0 0 50 0
0 Tm /TT1.0 1 Tf ( ) Tj ET Q 0 0 1 sc 161.04 707.28 m 259.68 707.28 l
259.68 707.04 l 161.04 707.04 l h f 0 0 0 sc q 0.24 0 0 0.24 90 695.28
cm BT 50 0 0 50 0 0 Tm /TT1.0 1 Tf ( ) Tj ET Q Q
```

FIG. 8—*The ASCII text stream produced when the binary stream in Fig. 7 is decompressed. (Line breaks have been added for legibility but do not occur in the source document.)*

the email address, one that was Type EF in the browser cache and one that was Type PNF in the swap space.

To distinguish trace evidence that can be recovered through conventional means from that which cannot, we assume that conventional techniques can recover any feature found in a recoverable file (types PF and EF), and that unencoded features can be recovered from raw sectors without the need for decompression or decoding (type PNF). Both of these assumptions overstate the capability of modern tools, resulting in results that are more conservative than they might otherwise be.

Experimental Design

This experiment relies on *bulk_extractor*, a research tool can scan and extract digital trace evidence in binary data using a variety of recognition approaches such as regular expressions. The tool can also detect data that are compressed or otherwise encoded with a variety of algorithms. A list of supported encodings appears in Table 2. Such data, when found, are decompressed or decoded and then reprocessed by both trace evidence scanners and the encoding detectors, a technique called *recursive*

TABLE 1—*A taxonomy of the conditions that digital trace evidence may be found on a forensic disk image.*

	Condition	Example
PF	Plain in File	An email address in the file *word1.doc*.
EF	Encoded in File	An email address in the file *word1.docx*.
PNF	Plain Not in File	An email address from Case 1 in which the file has been deleted and both the file system metadata and a portion of the file have been overwritten.*
ENF	Encoded Not in File	An email address from Case 2 in which the file has been deleted and both the file system metadata and a portion of the file have been overwritten.

*Swap and hibernation files are treated as NF space.

analysis. Thus, *bulk_extractor* implements both optimistic decompression and decoding as described in the introduction.

The *bulk_extractor* places the trace evidence that it recovers into a specially formatted text file that includes the offset of the each trace evidence item, the item itself, and the context in which the item was found. These items are called *features*, and the file is called a *feature file*. In the event that the item was compressed, the file includes the offset of the compressed data, the compression algorithm, and the offset within the decompressed data stream. The *bulk_extractor* uses the term "*forensic path*" to describe such features. Figure 9 shows examples of both a plain text and a compressed feature.

The *bulk_extractor* can identify and extract a wide variety of evidence types, including email addresses, credit card numbers, and URLs. This article only considers email addresses, but the findings should be relevant for all kinds of trace evidence.

A tool called *identify_filenames.py*, distributed with *bulk_extractor*, can associate each feature in the feature file with the file from which the file was extracted. The tool operates by analyzing the disk image and determining the location of each file. Next, the tool reads the feature file and, for each feature, determines whether the location of the feature on the media corresponds to an existing file. Finally, the tool creates an *annotated feature file* which includes the offset of each feature, the feature, the context, the file name in which the feature was found, and the MD5 hash value of the file.

Existing digital forensic tools can detect the presence of trace evidence such as email addresses that are in files of types PF, EF, and PNF; they generally cannot display data in type ENF. The question that this article seeks to answer, then, is whether or not there is a significant presence of trace evidence on digital media that is only present in the Type ENF form.

TABLE 2—*The kinds of encodings that can be decoded by bulk_extractor, and the amount of context required for the decoding.*

Encoding	Requires
GZIP	The beginning of a zlib-compressed stream
BASE64	The beginning of a BASE64-encoded stream
HIBER	Any fragment of a hibernation file can generally be decompressed, as each Windows 4K page is separately compressed and the beginning of each compressed page in the hibernation file is indicated by a well-known sequence
PDF	Any PDF stream compressed with ZLIB bracketed by *stream* and *endstream*
ZIP	The local file header of a ZIP-file component

To perform this analysis for a piece of media, the media is processed with both the *bulk_extractor* and the *identify_filenames.py* programs. The resulting annotated feature file is processed using an analysis program that was specially written for this study. The program makes a list of each email address and determines whether the email address is ever present in a file and ever present without encoding. The following rules are used for the determination:

Type PF: Plain in File Features that have a purely numerical offset (e.g., "12345") and an filename (e.g., "/Windows/System32/User.DLL").

Type EF: Encoded in File Features that have a forensic path containing a compression or other encoding method (e.g., "56789-GZIP-123") and an identified file name.

Type PNF: Plain Not in File Features that have a purely numerical offset but no filename.

Type ENF: Encoded Not in File Features that have a forensic path containing a compression method and no identified file name.

Email addresses that are only present in the ENF form are then tabulated.

Source Media

This experiment used 1646 drive images from the Real Data Corpus (15), a research corpus derived from several thousand hard drives, memory cards, CD-ROMs, DVDs, USB memory sticks, and cell phones procured from second-hand computer stores, open-air markets, and other locations in eight countries between 2005 and 2013. In each case, a *physical image* of the drive was made by copying the data sector-by-sector from the source device to an image file. Images were created with FTK Imager (16) and ewfacquire (17). The images were stored at several locations on a high-capacity storage array.

Each image was processed with *bulk_extractor*, *identify_filenames.py*, and the postprocessing program described above. For each drive, a list of the email addresses and each of the encoding types was created. Statistics were tabulated for the total number of email addresses on each drive and the number of email addresses that were present as Type ENF and the encoding algorithm for each of these addresses. Summary statistics were then created for all of the drives. Because the drives are considered separately, an email address that appears as type PF and ENF on the same drive will not be counted. However, an email address that appears as type PF on a first drive and type ENF on a second drive will be counted as a single-encoded email address.

Results

This section first reviews the summary results of all the ways that ENF email addresses were found on the subject drives. It then analyzes a variety of representative EF and the ENF email

```
392175418              WindowsXP@gn.microsoft.com   Name=WindowsXP@gn.microsoft.com\015\012
3772517888-GZIP-28322  user@company.com             onterey-<nobr>user@company.com</nobr>
```

FIG. 9—*Two lines of output from the bulk_extractor program. Each line represents a piece of digital trace evidence extracted from a piece of digital piece. The first column indicates the offset of the evidence (in decimal), the second column indicates the trace evidence itself, and third column shows the local context in which the trace evidence was found. In the case of the second line, the email address user@company.com was found inside a block of data that first needed to be decompressed. The compressed region was found at byte offset 3,772,517,888 and was compressed with the GZIP compression algorithm. The email address was found at an offset of 28,322 bytes from the start of the compressed region.*

addresses to determine the reason that the email addresses were present in the particular encoding.

Summary Results

A total of 961 drives were found to have email addresses in any of the four forms presented earlier; there were 1351 distinct email addresses per drive on average, with a minimum of 0 and a maximum of 178,201 on drive 1044. Detailed results can be found in Table 3.

By far, the majority of email addresses in this set were plain but not present in any file (type PNF). These email addresses can be recovered with traditional text processing operations such as carving and string search. Significant numbers of email addresses were found in files as well (types PF and EF). These can be recovered with traditional forensic file processing.

However, consistent with this article's hypothesis, a significant number of encoded email addresses were found in nonfile space of many drives. Email addresses were found encoded using BASE64, GZIP, PDF, HIBER (Microsoft XPress), and ZIP algorithms. Furthermore, while a majority of the encoded email addresses found were encoded with a single algorithms, thousands of distinct email addresses were found on dozens of drives that were sequentially encoded with multiple algorithms. Such email addresses can only be recovered through the kind of recursive processing exhibited by the *bulk_extractor* program.

Although there are a wide variety of ENF email addresses present in our sample, it was clear from a manual examination of the email addresses that not all were of equal forensic importance. Some encodings are clearly the result of email addresses included in software distributions, while others are clearly the result of user-generated content. In general, forensic examiners are more concerned with user-generated content, but there may be cases where the software present on subject media is equally important.

The following sections review representative email addresses that were found in various encodings in the corpus. In some cases, the email addresses were found to come from publicly available documents. In those cases, example output from *bulk_extractor* is provided.

BASE64 Encoding

BASE64 is scheme that allows arbitrary binary sequences to be encoded with printable letters (both uppercase and lowercase), numerals, the plus sign, and a forward slash (18). BASE64 is widely used to encode Internet email attachments and SSL certificates, as well as to represent binary information inside XML documents.

A manual examination of representative NF BASE64-encoded email addresses in the corpus found that the majority were from email messages. In many cases, the context was clearly HTML. Much of the HTML was formatted in lines terminated with CR/LF pairs (the standard end-of-line encoding for email messages), rather than bare LF characters (common for web pages downloaded by HTTP). In some cases, the email addresses appear to be taken from Email headers.

We hypothesize that BASE64-encoded HTML resulted from email messages that were sent as attachments. Such encoding is common with modern email clients such as Microsoft Outlook and Mozilla Thunderbird. Manual examination confirmed that the majority of email addresses encoded as BASE64 were privacy-sensitive user-generated content.

Base64-GZIP

The GZIP compression algorithm is commonly used by web servers to transparently compress web pages, and by software developers for distributing source code. Data that are BASE64-GZIP encoded were first compressed with GZIP and then with BASE64. This does not correspond to the way that web servers would encode data, but is how one would expect to find computer source code that is sent as an email attachment.

TABLE 3—*Summary statistics for the number of distinct email addresses found on each drive in each coding variant. A total of 1646 drives were examined, of which 961 were found to contain at least one email address.*

Coding	Drives with >1 Email Addrs	Total Distinct Email Addresses	Avg per Drive	Max per Drive	σ
(1) Plain in File	739	81,920	110	4206	253
(2) Encoded in File	355	19,711	55	5454	388
(3) Plain Not in File	860	195,605	2274	178,073	9248
(4) Encoded Not in File	474	165,481	349	59,376	2889
BASE64	54	219	4	50	7
BASE64-GZIP	2	64	32	37	5
BASE64-GZIP-GZIP	2	2	1	1	0
GZIP	234	66,195	282	9103	981
GZIP-BASE64	7	44	6	11	3
GZIP-GZIP	15	12,663	844	11,845	2944
GZIP-GZIP-BASE64	2	38	19	30	11
GZIP-GZIP-GZIP	4	58	14	38	14
GZIP-GZIP-ZIP	1	12	12	12	0
GZIP-PDF	5	38	7	30	11
GZIP-ZIP	6	49	8	30	9
HIBER	79	1433	18	217	44
HIBER-GZIP	1	2	2	2	0
PDF	162	2352	14	238	31
ZIP	388	85,252	219	59,369	3025
ZIP-BASE64	5	30	6	13	5
ZIP-BASE64-GZIP	2	65	32	38	5
ZIP-BASE64-GZIP-GZIP	2	2	1	1	0
ZIP-GZIP	14	261	18	132	34
ZIP-PDF	26	115	4	18	4
ZIP-ZIP	67	430	6	48	8
ZIP-ZIP-ZIP	3	9	3	6	2

A manual examination of email addresses and local context encoded with as BASE64-GZIP found that the data could be readily traced to publicly available software repositories. For example, drive il2-0027 contains this sequence:

```
path: 151336103-BASE64-5102-GZIP-375099
feature: schwartz@cs.tu-berlin.de
context: b'artin Schwartz schwartz@cs.tu-ber-
lin.de creator of "la'
```

Web searching reveals that the context is part of the README file for the Antiword software package.

Another example taken from the same disk:

```
path: 101814559-BASE64-0-GZIP-239191
feature: vajper@datorklubben.ml.org
context: b'Henrik Persson (vajper@datorklubben.
ml.org) brand:
```

Although this snippet appears to contain private PII, the text actually is a configuration file for the Philips VCR3 infrared remote control system and is part of the Linux Infrared Remote Control package available for download at http://lirc.sourceforge.net/remotes/philips/VCR3.

Additional manual exploration confirmed that the majority of the email addresses encoded BASE64-GZIP are from source code, many as evidenced by the fact that the email addresses are closely followed by a copyright claim.

GZIP Encoding

Our examination of GZIP-encoded email addresses in NF space found that they came from a wide variety of sources, including downloaded web pages, JSON objects, and open-source software. In general, the email addresses found in GZIP-encoded HTML and in close proximity to copyright statements appeared to be publicly available, while email addresses contained in JSON sequences appeared to be private information.

HIBER Encoding

The HIBER-encoding method is used exclusively by Windows-based computers to compress RAM before it is written to the HIBERFIL.SYS file when the computer goes into hibernation. As expected, email addresses encoded with HIBER in NF space take on all of the forms of email addresses that are found in system RAM. Although some email addresses are clearly from web pages and email messages, the majority seem to be embedded in some kind of binary data structure.

A significant number of the HIBER-encoded email identified by *bulk_extractor* were actually false positives—that is, they were character strings that appear to be email addresses but which were not, such as the string FRANCES@WWW.MS.

This study confirmed our assumption that HIBER-encoded data present on storage media are invariably result from data that was in RAM and was written to disk as part of a Windows hibernation. Thus, the same techniques used for analyzing data in RAM can be productively applied to analyzing these data as well. However, HIBER-encoded ENF data are likely to be older than data in RAM, as the pages were removed from the file system as a result of the hibernation file being deleted by the user or relocated as part of a file system defragmentation.

PDF Encoding

Many drives in the corpus contain email addresses in PDF streams. In the case of drives that had just a few PDF-encoded email addresses, these addresses appeared to come from computer documentation. For example, drive cn4-06 contains this feature which appears to be from a PDF describing how to configure an email client:

```
path: 1361681035-PDF-7
feature: yourname@provider.co.jp
context: mailto:yourname@provider.co.jp #
```

A few drives in the corpus contained hundreds or thousands of PDF-encoded email addresses. Manual inspection of these cases indicated that they were overwhelmingly email addresses corresponding to individuals and that the PDFs are not publicly available. Presumably, a user had generated a PDF file containing email addresses and that PDF file was later deleted and partially overwritten. As a result, PDF-encoded email addresses could contain case-relevant trace evidence if the original PDF was relevant to the case at hand. Because the email addresses were ENF, they would be missed by conventional forensic tools.

ZIP

Features are reported as being ZIP encoded if they were found in a fragment of a ZIP file. Although ZIP was originally developed as a compressed container file for archiving and distributing collections of files, today ZIP is also used for an astonishing number of applications, including Java byte code libraries, Android and iPhone applications, Microsoft and Open Office document files, and distributing source code. Not surprisingly, ENF email addresses found with ZIP encoding appear to come from a wide variety of sources.

Analysis of the ZIP-encoded email addresses finds that they are dominated by email addresses from Java libraries. For example, drive PS01-070 contains 450 copies of an email address and associated verbiage from the Apache Software License, presumably from a Java JAR file:

```
path: 3333029290-ZIP-1283
feature: apache@apache.org
context: please contact apache@apache.org. * *
5. Produ
```

ZIP-PDF

We hypothesized that email addresses that were found with ZIP-PDF encoding presumably came from PDF files containing email addresses that were bundled together into a ZIP archive, written to the storage device, and then partially overwritten. In an actual case, such trace evidence might be the result of data theft or exfiltration.

In our analysis, we found that the majority of the ZIP-PDF-encoded features appear to come from private documents. In a few cases, we found that the PDFs were publicly available. For example, disk il3-0184 contained the following feature:

```
path: 3076019889-ZIP-174034-PDF-2969
feature: SOAP4fun@the-beach.com
context: n t name I-want: SOAP4fun@the-beach.
com . 68
```

This feature was found with a web search and attributed to the *gSOAP 2.6.0 User Guide*; we downloaded a copy from `http://alien.cern.ch/cache/gsoap-2.6/soapcpp2/soapdoc2.pdf`.

In the majority of the cases, however, ZIP-PDF-encoded email addresses came from private documents.

ZIP-ZIP-ZIP Encoding

The majority of the email addresses that were encoded ZIP-ZIP-ZIP were from documents that accompanied various WinZip software distributions, including the WinZip 5.5 Registration Form and the WinZip 8.1 Registration and Order Information form. We hypothesize that this text file was included in a ZIP archive that was part of a WinZip distribution that itself was compressed into a WinZip archive and redistributed as a self-extracting ZIP archive. We did not find a single case in which a ZIP-ZIP-ZIP-encoded email address contained user-generated content.

Discussion

In our sample of 961 storage devices acquired around the world, roughly one-in-four contained email addresses that were not in plain encoded but did not reside in a file. Many of these email addresses came from user-generated content. Because these email addresses were encoded, they were not in plain text and, as such, would be invisible to the majority of today's digital forensic tools.

Drives in the sample came from the Real Data Corpus and were restricted to drives that already had at least one email address. Filtering the RDC in this manner avoided including drives in the count that only contained JPEGs or that had been properly sanitized before being sold on the secondary market.

We found that some kinds of encoding formats were used almost exclusively for email addresses originating in user-generated content. For example, email addresses that were encoded with BASE64 largely originated in email attachments, and specifically other email messages that were forwarded as attachments. Of our sample, 54 drives had more than one email address that was encoded with the BASE64 algorithm and not in a file (ENF), or roughly 1 in 30. Likewise, email addresses with GZIP were likely to be from user-generated content; 234 drives contained ENF email addresses.

Suppressing Nonuser-generated Content

It is clear that many of the email addresses found by the *bulk_extractor* do not result from user-generated content and, as such, may have limited role in many investigations. For example, the email addresses iemand@microsoft.com and monnom@msn.com were found on many disks, but *iemand* is Dutch for *someone* and *mon nom* is French for *my name*, both of which are reasonable-looking test email addresses for native speakers of those languages.

One approach for suppressing email addresses and other features that are not user-generated is to build stop lists of features that appear in default software distributions. A detailed approach for doing this with context-specificity appears in (4).

Conclusion

Digital forensics is a powerful tool that is widely used by different investigators for many different purposes. Some investigations are limited in scope, the kinds of tools that may be used, and the information that may be examined. For example, in some cases, investigators are free to search unallocated space, whereas in others investigators are limited to examining allocated files. In circumstances where the investigation has unlimited access to subject media, investigators who do not consider encoded email addresses that are in nonfile space are potentially missing important trace evidence.

Acknowledgments

I gratefully acknowledge Josiah Dykstra, Kyle Gorak, Aubin James Heffernan, and Carolina Zarate for their useful feedback on previous versions of this article.

References

1. Garfinkel S, Shelat A. Remembrance of data passed. IEEE Secur Priv 2003;1(1):17–27.
2. Carrier B. File system forensic analysis. Upper Saddle River, NJ: Pearson Education, 2005.
3. Beverly R, Garfinkel S, Cardwell G. Forensic carving of network packets and associated data structures. Proceedings of the Eleventh Annual DFRWS Conference; 2011 Aug 1-3; New Orleands, LA. Elsevier, 2011.
4. Garfinkel S. Stream-based digital media forensics with *bulk_extractor*. Comput Secur 2013;32:57–72.
5. Suiche M. Windows hibernation file for fun 'n' profit. In: Black hat, 2008; http://sebug.net/paper/Meeting-Documents/BlackHat-USA2008/BH_US_08_Suiche_Windows_hibernation.pdf (accessed December 3, 2011).
6. Garfinkel S, Migletz J. New XML-based files: implications for forensics. IEEE Security & Privacy Magazine 2009 March/April;7(2):38–44; http://simson.net/clips/academic/2009.IEEE.DOCX.pdf.
7. Digital Assembly. Adroit photo forensics, 2011; http://digital-assembly.com/ (accessed December 3, 2011).
8. Grenier C. Photorec, 2011; http://www.cgsecurity.org/wiki/PhotoRec (accessed December 3, 2011).
9. Microsoft Corp. [ms-doc]: Word (.doc) binary file format, February 11 2013; http://msdn.microsoft.com/en-us/library/cc313153 (accessed February 23, 2013).
10. Office open xml, ecma-376, and iso/iec 29500, February 2011; http://msdn.microsoft.com/en-us/library/office/gg607163 (accessed February 23, 2013).
11. Adobe Systems Incorporated. PDF reference and adobe extensions to the pdf specification, 2013.
12. Roelofs G, Adler M. zlib: a massively spiffy yet delicately unobtrusive compression library (also free, not to mention unencumbered by patents), May 2, 2012; http://www.zlib.net (accessed February 23, 2013).
13. Oracle. Oracle outside in technology, 2013; http://www.oracle.com/us/technologies/embedded/025613.htm (accessed February 23, 2013).
14. Katz P. APPNOTE.TXT — .ZIP File Format Specification. Technical report, PKWare, Inc., September 28 2007; http://www.pkware.com/documents/casestudies/APPNOTE.TXT.
15. Garfinkel SL, Farrell P, Roussev V, Dinolt G. Bringing science to digital forensics with standardized forensic corpora. Proceedings of the 9th Annual Digital Forensic Research Workshop; 2009 Aug 17-19; Montreal, Canada. Elsevier, 2009.
16. Access Data. Forensic toolkit (FTK), 2011; http://accessdata.com/products/computer-forensics/ftk (accessed December 3, 2011).
17. Metz J. libewf: Project info, 2008; http://sourceforge.net/projects/libewf/ (accessed December 3, 2011).
18. Josefsson S. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), October 2006; http://www.ietf.org/rfc/rfc4648.txt.

Additional information and reprint requests:
Simson L. Garfinkel, Ph.D.
Naval Postgraduate School
Compuer Science
1186 North Utah Street
Arlington, VA 22201-4758
E-mail: simsong@acm.org