Overview:

- What is bulk_extractor?
- What can it do?
- How does it work?
- How do I run it?
- What’s new in version 1.5?
Introducing bulk_extractor
bulk_extractor is a stream-based disk forensics tool. It scans the media and extracts recognizable content.

1. Read all of the blocks in order.
2. Look for information that might be useful.
3. Identify & extract what's possible in a single pass.

3 hours, 20 min to read the data
Before bulk_extractor, this was done with strings & grep

```
$ strings diskimage.dd | grep '[a-z]+*@[a-z0-9]+'
```

Problems with this approach:
- Slow (not parallelized)
- Many false positives
- Each ‘search’ requires a complete scan of the device
- Misses encoded data.
bulk_extractor improves on strings & grep

Finds more kinds of data:
- Structured text (email addresses, URLs, etc)
- Structured binary data (Microsoft Windows PE files, LNK files, etc)

Finds data in many kinds of situations:
- Compressed & encoded data.
- Recursive re-analysis

Expandable:
- Easy to add new features.
“Encoded data” must frequently be \textit{decoded} to be recognized.

Compression removes redundancies in data:

\begin{verbatim}
5859 5a40 636f 6d70 616e 792e 636f 6d 4142 4300 636f 6d70 616e 792e 636f 6d 4445 4600 636f 6d70 616e 792e 636f 6d
XYZ@company.com
ABC@company.com
DEF@company.com
\end{verbatim}

Compressed with “gzip:"

\begin{verbatim}
1f8b 0800 0000 0000 0020 0203 8b88 8c72 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce 48ce
......
cf2d 48cc abd4 03d2 0a8e 4ece 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757 287c 1757
3714 3e00 b455 c1c5 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000 3000 0000
\end{verbatim}

Compressed email addresses do not “look” like email addresses!

—\textit{Forensic tools must “optimistically” decompress data to search for email addresses.}
Programs encode data in many ways.

FIG. 1—A Microsoft Word file containing a single sentence followed by a blank line.
Word’s .doc format stores plain text (UTF-8 and UTF-16)

000000a00: 4f6e 6520 7477 6f20 7468 7265 6520 1320 One two three .
000000a10: 4859 5045 524c 494e 4b20 226d 6169 6c74 HYPERLINK ‘‘mailto
000000a20: 6f3a 7573 6572 4063 6f6d 7061 6e79 2e63 o:user@company.c
000000a30: 6f6d 2220 1475 7365 7240 636f 6d70 616e om’’.user@compan
000000a40: 792e 636f 6d15 2066 6f75 7220 6669 7665 y.com. four five
000000a50: 2073 6978 2e0d 0d00 0000 0000 0000 0000 six.......... six...........
000000a60: 0000 0000 0000 0000 0000 0000 0000 0000 ............... six............
000000a70: 0000 0000 0000 0000 0000 0000 0000 0000 ............... six............
Word’s .docx format stores content as compressed XML

```
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.xml
000009c0: 6ca4 55db 729b 3010 7def 4cff 81d1 7b0c 1.L.U.r.O.].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
```

Uncompress

```
<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>
```
PDFs generated by Word are compressed PDF streams
Encoded data may be in files or between files.

Folders.pst

Presentation.pptx

Sequestration.docx

Mother.JPG
EnCase & FTK use Oracle’s “Outside In” to extract text.

Outside In will extract text from:
- Word .doc
- Word .docx
- PDFs made by Word
- 500+ other file formats
Outside In won’t extract text from non-file (“bulk”) data.

- The entire file may not be present.

Examples:
- Compressed — zlib (gzip, ZIP), RAR, Windows Hibernation (Microsoft Xpress)
- Encoded — BASE64
- Obfuscated — ROT13, XOR(255)
bulk_extractor splits the disk into 16M “pages” (blocks) and processes each page independently.

This finds obvious email addresses in bulk data:

```
XYZ@COMPANY.COM
ABC@company.com
DEF@company.com
```

```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399 ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5 ....i/XG...S,..
b061 b64c 721d 864b 90b6 b55f bb04 735c Pa.Lr..K..._..s\n9448 6730 5453 df64 813e b603 5795 2242 .Hg0TS.d.>..W."B
e9c8 7454 7322 7cdc b60f 97af 2f64 2728 ..tTs"|....../d'
3cfb 84bd 2a84 2dfe 50ea 5935 c349 1513 <XYZ@COMPANY.COM
a9e9 e92c a3f8 6e46 0530 8a88 c7a2 5d2b ...,...nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b ..w.....7......G..
```
bulk_extractor examines every byte to see if it is the beginning of an “encoded” region. Once the region is found, it’s decoded, then processed.

This “optimistic” approach also recovers data from fragments of files.
bulk_extractor has three phases of operation:
Feature extraction; histogram creation; post processing

Output is a directory containing:
- feature files; histograms; carved objects
- Mostly in UTF-8; some XML
- Can be bundled into a ZIP file and processed with bulk_extractor_reader.py
bulk_extractor is run from the command line and creates a directory of “feature files” and carved results.

Command line:

```bash
$ bulk_extractor -o output_dir INPUT.E01
```

$ ls -l out-team

```
total 228
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 aes_keys.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 alerts.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ccn.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ccn_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ccn_track2.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ccn_track2_histogram.txt
-rw-r-----+ 1 simsong  staff 18918 Apr 21 13:15 domain.txt
-rw-r-----+ 1 simsong  staff  854 Apr 21 13:15 domain_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 elf.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 email.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 email_domain_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 email_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ether.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ether_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 exif.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 find.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 find_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 gps.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ip.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 ip_histogram.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 jpeg_carved.txt
-rw-r-----+ 1 simsong  staff  6646 Apr 21 13:15 json.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 kml.txt
-rw-r-----+ 1 simsong  staff 147729 Apr 21 13:15 pii.txt
-rw-r-----+ 1 simsong  staff  313 Apr 21 13:15 pii_teamviewer_from.txt
-rw-r-----+ 1 simsong  staff 0 Apr 21 13:15 rar.txt
```

“0” means scanner ran, nothing found

teamviewer found
BEViewer: GUI runs on Windows, Mac & Linux
Launches bulk_extractor; views results

Uses bulk_extractor to decode encoded data.

The viewer can run on an existing report.
“Feature files” contain the results of scanners.

Line-by-line output.

```
# Filename: /corp/nps/drives/nps-2009-m57-patents/charlie-2009-12-11.E01
# Feature-Recorder: telephone
# Feature-File-Version: 1.1
...
6489225486 (316) 788-7300 Corrine Porter (316) 788-7300,,,,,,Phase I En
6489230027 620-723-2638 ,,,,Dan Hayse - 620-723-2638,,,,,,Phase I En
6489230346 620-376-4499 Bertha Mangold -620-376-4499,,,,,,Phase I En
```

- Designed for easy processing by grep, Notepad++, python or perl
  - “Loosely ordered.”
  - UTF-8 clean (non-UTF-8 characters are escaped with Python-notation)
The “context” helps you decide if the hit is valid or not.

These are not valid credit card numbers:

- 85364212076 346646148754362 /photo.php?fbid=346646148754362&set=a.345157228
- 85480391075 346344211011114 <t~x~\x80\x8A\x85vvx\x8A\x8Cm;/346344211011114/U}
d1rzyn\x83\x89\x8A\x8B\x87\x89\x88
- 87433290505 349446478403970 /photo.php?fbid=349446478403970&set=a.182950211
- 87434352827 347642328585048 /photo.php?fbid=347642328585048&set=t.100000231
- 87435156790 371055846243033 /photo.php?fbid=371055846243033&set=o.221006364
- 87435883832 347439688656505 /photo.php?fbid=347439688656505&set=a.292909134

These probably are:

- 18462439451 4519********3362 \x0A\x0D\x0ALynn:\x0D\x0ACard:
  4519********3362\x0D\x0APassword: rosi
- 18462439594 4519********3362 DINEROLYNN:\CARD:
  4519********3362PASSWORD:ROSIETO

CCNs are validated with the Luhn algorithm & heuristics.
The “offset” also indicates how the data were decoded. We call this the “forensic path.”

- “-GZIP-” indicates that data was decompressed
Decoding transformations can be stacked.

This output comes from a GZIP stream in a Windows Hibernation File.

• JSON object downloaded from Facebook by compressed HTTP
  • In RAM, written to HIBER on disk when the system went into sleep.
    —All same from same HIBER section & same GZIP object.
The file “report.xml” is an XML log of the processing.

```
-rw-r-----+ 1 simsong  staff   32766 Apr 21 13:15 report.xml
```

report.xml contains:

- Invoking command:
  
  ```
  $ grep command out-M1234/report.xml
       <command_line>src/bulk_extractor -Z -o out-M1234 M1234.img</command_line>
  $ 
  ```

- Compiler used to compile executable & linked libraries.
- System on which executable ran.
- Scanners that ran
- Statistics
- Performance counters
“tests/regress.py --analyze”
will report time spent by each scanner

% python tests/regress.py --analyze ~/IN10-0512-fb
Analyze /home/simsong/IN10-0512-fb
bulk_extractor version: 1.5.0-alpha5
Image filename: /corp/nus/drives/IN/IN10-0512/IN10-0512.E01
Scanner paths by time and calls

<table>
<thead>
<tr>
<th>name</th>
<th>calls</th>
<th>sec</th>
<th>sec/call</th>
<th>% total</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMAIL</td>
<td>1813</td>
<td>4778.21</td>
<td>2.6355</td>
<td>23.03%</td>
</tr>
<tr>
<td>ZIP</td>
<td>1813</td>
<td>3110.51</td>
<td>1.7157</td>
<td>14.99%</td>
</tr>
<tr>
<td>NET</td>
<td>1813</td>
<td>2718.79</td>
<td>1.4996</td>
<td>13.10%</td>
</tr>
<tr>
<td>ACCTS</td>
<td>1813</td>
<td>2134.94</td>
<td>1.1776</td>
<td>10.29%</td>
</tr>
<tr>
<td>AES</td>
<td>1813</td>
<td>1656.88</td>
<td>0.9139</td>
<td>7.99%</td>
</tr>
<tr>
<td>ZIP-NET</td>
<td>42060</td>
<td>832.95</td>
<td>0.0198</td>
<td>4.01%</td>
</tr>
<tr>
<td>ZIP-EMAIL</td>
<td>42060</td>
<td>745.83</td>
<td>0.0177</td>
<td>3.59%</td>
</tr>
<tr>
<td>HTTPLOGS</td>
<td>1813</td>
<td>483.83</td>
<td>0.2669</td>
<td>2.33%</td>
</tr>
<tr>
<td>FIND</td>
<td>1813</td>
<td>439.82</td>
<td>0.2426</td>
<td>2.12%</td>
</tr>
<tr>
<td>ZIP-ACCTS</td>
<td>42060</td>
<td>374.70</td>
<td>0.0089</td>
<td>1.81%</td>
</tr>
<tr>
<td>ZIP-AES</td>
<td>42060</td>
<td>361.45</td>
<td>0.0086</td>
<td>1.74%</td>
</tr>
<tr>
<td>BASE64</td>
<td>1813</td>
<td>338.88</td>
<td>0.1869</td>
<td>1.63%</td>
</tr>
<tr>
<td>WINLNK</td>
<td>1813</td>
<td>309.98</td>
<td>0.1710</td>
<td>1.49%</td>
</tr>
<tr>
<td>RAR</td>
<td>1813</td>
<td>298.12</td>
<td>0.1644</td>
<td>1.44%</td>
</tr>
<tr>
<td>WINPE</td>
<td>1813</td>
<td>265.66</td>
<td>0.1465</td>
<td>1.28%</td>
</tr>
<tr>
<td>HIBER</td>
<td>1813</td>
<td>265.19</td>
<td>0.1463</td>
<td>1.28%</td>
</tr>
<tr>
<td>EXIF</td>
<td>1813</td>
<td>185.51</td>
<td>0.1023</td>
<td>0.89%</td>
</tr>
</tbody>
</table>
Inside bulk_extractor
bulk_extractor: architectural overview

Written in C++, GNU flex and Java (GUI)
- Command-line tool.
- Linux, MacOS, Windows (compiled with mingw)
- BEViewer command-line tool and views results
- Overall size:
  - 23,911 lines C++
  - 1,654 lines GNU Flex
  - 17,861 lines of Java

Key Features:
- “Scanners” look for information of interest in typical investigations.
- Recursively re-analyzes compressed data.
- Results stored in “feature files”
- Multi-threaded

http://www.nps.edu/
202-555-1212
user@domain.com
http://www.nps.edu/
202-555-1212
user@domain.com
bulk_extractor: system diagram
image processing
C++ iterator handles disks, images and files

Works with multiple disk formats.
- E01
- raw
- split raw
- individual disk files
- (AFF)

We chop the 1TB disk into 65,536 x 16MiB “pages” for processing.
The “pages” overlap to avoid dropping features that cross buffer boundaries.

The overlap area is called the *margin*.

- Each sbuf can be processed in parallel — they don’t depend on each other.
- Features that start in the page but end in the margin are *reported*.
- Features that start in the margin are *ignored* (we get them later)
  — Assumes that the feature size is smaller than the margin size.
  — *BE version 1.5 margin: 4MB (tunable)*

Entire system is automatic:

- Image_process iterator makes `sbuf_t` buffers.
- Each buffer is processed by every scanner
- Features are automatically combined.
bulk_extractor has many scanners. Each scanner runs sequentially on all the data.

Scanners can be turned on or off
- Use for tuning & debugging
  — *Turn off scanners you don’t need.*
  — *Turn off scanners if you get a crash.*

Some scanners are recursive.
- *e.g.* scan_zip will find zlib-compressed regions
- An *sbuf* is made for the decompressed data
- The data is re-analyzed by the other scanners
  — *This finds email addresses in compressed data!*

Some scanners can “carve”

Recursion used for:
- Decompressing ZLIB, Windows HIBERFILE,
- Extracting text from PDFs
Scanners process each page and extract features

scan_email is the email scanner.
- inputs: `sbuf` objects

outputs:
- `email.txt`
  - Email addresses
- `rfc822.txt`
  - Message-ID
  - Date:
  - Subject:
  - Cookie:
  - Host:
- `domain.txt`
  - IP addresses
  - host names
The feature recording system saves features to disk.

**Feature Recorder** objects store the features.
- Scanners are given a `(feature_recorder *)` pointer
- Feature recorders are *thread safe.*

Features are stored in a *feature file*:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Email Address</th>
<th>Feature Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>48198832</td>
<td><a href="mailto:domexuser2@gmail.com">domexuser2@gmail.com</a></td>
<td>tocol&gt;<em><strong>&lt;name&gt;<a href="mailto:domexuser2@gmail.com">domexuser2@gmail.com</a>/Home&lt;/name&gt;</strong></em></td>
</tr>
<tr>
<td>48200361</td>
<td><a href="mailto:domexuser2@live.com">domexuser2@live.com</a></td>
<td>tocol&gt;<em><strong>&lt;name&gt;<a href="mailto:domexuser2@live.com">domexuser2@live.com</a>&lt;/name&gt;</strong></em>&lt;pass</td>
</tr>
<tr>
<td>48413829</td>
<td><a href="mailto:siege@preoccupied.net">siege@preoccupied.net</a></td>
<td>siege) O'Brien <a href="mailto:siege@preoccupied.net">siege@preoccupied.net</a> hp://meanwhi</td>
</tr>
<tr>
<td>48481542</td>
<td><a href="mailto:danilo@gnome.org">danilo@gnome.org</a></td>
<td>Danilo __egan <a href="mailto:danilo@gnome.org">danilo@gnome.org</a>_Language-Team:</td>
</tr>
<tr>
<td>48481589</td>
<td><a href="mailto:gnom@prevod.org">gnom@prevod.org</a></td>
<td>: Serbian (sr) <a href="mailto:gnom@prevod.org">gnom@prevod.org</a>_MIME-Version:</td>
</tr>
<tr>
<td>49421069</td>
<td><a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a></td>
<td>server2.name&quot;, &quot;<a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a>&quot;);__user_pref(&quot;</td>
</tr>
<tr>
<td>49421279</td>
<td><a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a></td>
<td>er2.userName&quot;, &quot;<a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a>&quot;);__user_pref(&quot;</td>
</tr>
<tr>
<td>49421608</td>
<td><a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a></td>
<td>tpl1.username&quot;, &quot;<a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a>&quot;);__user_pref(&quot;</td>
</tr>
</tbody>
</table>
Features can also be stored in a SQLite3 database*

Each “feature table” that stores:

- offset bytes from start of disk (integer)
- path forensic path (string)
- feature_eutf8 feature, escaped UTF8 (string)
- feature_utf8 feature, pure UTF8 (string)
- context_eutf8 escaped UTF8 (string)

```
sqlite> select * from f_email limit 10;

<table>
<thead>
<tr>
<th>offset</th>
<th>path</th>
<th>feature_eutf8</th>
<th>feature_utf8</th>
</tr>
</thead>
<tbody>
<tr>
<td>42716701</td>
<td>42716701-BASE64-60</td>
<td><a href="mailto:pki@microsoft.com">pki@microsoft.com</a></td>
<td><a href="mailto:pki@microsoft.com">pki@microsoft.com</a></td>
</tr>
<tr>
<td>42716701</td>
<td>42716701-BASE64-396</td>
<td><a href="mailto:pki@microsoft.com">pki@microsoft.com</a></td>
<td><a href="mailto:pki@microsoft.com">pki@microsoft.com</a></td>
</tr>
<tr>
<td>24900678</td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td>\x028\x0...</td>
</tr>
<tr>
<td>26735686</td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td>\x028\x0...</td>
</tr>
<tr>
<td>32597062</td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td><a href="mailto:grafta@bl.com">grafta@bl.com</a></td>
<td>\x028\x0...</td>
</tr>
<tr>
<td>50392739</td>
<td><a href="mailto:inet@microsoft.com">inet@microsoft.com</a></td>
<td><a href="mailto:inet@microsoft.com">inet@microsoft.com</a></td>
<td><a href="mailto:inet@microsoft.com">inet@microsoft.com</a></td>
</tr>
<tr>
<td>51781228</td>
<td><a href="mailto:dbaron@dbaron.org">dbaron@dbaron.org</a></td>
<td><a href="mailto:dbaron@dbaron.org">dbaron@dbaron.org</a></td>
<td><a href="mailto:dbaron@dbaron.org">dbaron@dbaron.org</a></td>
</tr>
<tr>
<td>51788157</td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
</tr>
<tr>
<td>51789901</td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
<td><a href="mailto:bzbarsky@mit.edu">bzbarsky@mit.edu</a></td>
</tr>
<tr>
<td>51791829</td>
<td><a href="mailto:roland.mainz@informatik.med.uni-giessen.d">roland.mainz@informatik.med.uni-giessen.d</a>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Enable with command line option:

```
-S write_feature_sqlite3=YES
```
Feature histograms are created at the end of processing. They are a powerful tool for understanding relations.

Email address histogram allows us to rapidly determine:

- Drive’s primary user
- User’s organization
- Primary correspondents
- Other email addresses

Drive #51
(Anonymized)

<table>
<thead>
<tr>
<th>Email Address</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:ALICE@DOMAIN1.com">ALICE@DOMAIN1.com</a></td>
<td>8133</td>
</tr>
<tr>
<td><a href="mailto:BOB@DOMAIN1.com">BOB@DOMAIN1.com</a></td>
<td>3504</td>
</tr>
<tr>
<td><a href="mailto:ALICE@mail.adhost.com">ALICE@mail.adhost.com</a></td>
<td>2956</td>
</tr>
<tr>
<td><a href="mailto:JobInfo@alumni-gsb.stanford.edu">JobInfo@alumni-gsb.stanford.edu</a></td>
<td>2108</td>
</tr>
<tr>
<td><a href="mailto:CLARE@aol.com">CLARE@aol.com</a></td>
<td>1579</td>
</tr>
<tr>
<td><a href="mailto:DON317@earthlink.net">DON317@earthlink.net</a></td>
<td>1206</td>
</tr>
<tr>
<td><a href="mailto:ERIC@DOMAIN1.com">ERIC@DOMAIN1.com</a></td>
<td>1118</td>
</tr>
<tr>
<td><a href="mailto:GABBY10@aol.com">GABBY10@aol.com</a></td>
<td>1030</td>
</tr>
<tr>
<td><a href="mailto:HAROLD@HAROLD.com">HAROLD@HAROLD.com</a></td>
<td>989</td>
</tr>
<tr>
<td><a href="mailto:ISHMAEL@JACK.wolfe.net">ISHMAEL@JACK.wolfe.net</a></td>
<td>960</td>
</tr>
<tr>
<td><a href="mailto:KIM@prodigy.net">KIM@prodigy.net</a></td>
<td>947</td>
</tr>
<tr>
<td><a href="mailto:ISHMAEL-list@rcia.com">ISHMAEL-list@rcia.com</a></td>
<td>845</td>
</tr>
<tr>
<td><a href="mailto:JACK@nwlink.com">JACK@nwlink.com</a></td>
<td>802</td>
</tr>
<tr>
<td><a href="mailto:LEN@wolfenet.com">LEN@wolfenet.com</a></td>
<td>790</td>
</tr>
<tr>
<td><a href="mailto:natcom-list@rcia.com">natcom-list@rcia.com</a></td>
<td>763</td>
</tr>
</tbody>
</table>
Histograms can be based on regular expressions to extract strings from feature files.

**Simple histogram based on “pure” feature:**

- n=579 domexuser1@gmail.com
- n=432 domexuser2@gmail.com
- n=340 domexuser3@gmail.com
- n=268 ips@mail.ips.es
- n=252 premium-server@thawte.com
- n=244 CPS-requests@verisign.com
- n=242 someone@example.com

**Based on regular expression extraction:**

- For example, extract search terms with \*search.*q=(.*)
  - n=18 pidgin
  - n=10 hotmail+thunderbird
  - n=3 Grey+Gardens+cousins
  - n=3 dvd
  - n=2 %TERMS%
  - n=2 cache:
  - n=2 p
  - n=2 pi
  - n=2 pid
  - n=1 Abolish+income+tax
  - n=1 Brad+and+Angelina+nanny+help
  - n=1 Build+Windmill
  - n=1 Carol+Alt
Use the histograms to get an overview of the media.

Histograms created by default in bulk_extractor 1.5:

- ccn_histogram — credit card numbers
- ccn_track2_histogram — track2 information
- domain_histogram — All domains found on the drive
- email_domain_histogram — Domains from email addresses
- email_histogram — Email addresses
- ip_histogram — IP addresses (frequently false positives)
- pii_teamviewer* — TeamViewer IDs
- telephone_histogram — Telephone numbers
- url_facebook — Facebook URLs
- url_histogram — JURLs
- url_microsoft_live — Microsoft Live URLs
- url_searches — What people searched for
- url_services — Domains from URLs
url_searches.txt is a list of search URLs from media.

```
$ more out-domexusers-baseline/url_searches.txt
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.5.0-beta1 ($Rev: 10844 $)
# Feature-Recorder: url
# Histogram-File-Version: 1.1
n=18    pidgin
n=10    hotmail+thunderbird
n=7     %s
n=3     Grey+Gardens+cousins
n=3     dvd
n=2     %TERMS%
n=2     p
n=2     pi
n=2     pid
n=1     Abolish+income+tax
n=1     Brad+and+Angelina+nanny+help
n=1     Build+Windmill
n=1     Carol+Alt
n=1     DVD
n=1     Don+Quixote
n=1     Ivanka+Trump
n=1     John+Updike
n=1     Kate+Hudson
n=1     Obama+McCain+Palin+Biden
n=1     Patrick+Swayze
n=1     StarCaps
```
BE1.5 can also store histograms in SQLite3*

```sql
sqlite> select * from h_email order by count desc limit 10;
<table>
<thead>
<tr>
<th>count</th>
<th>feature_utf8</th>
</tr>
</thead>
<tbody>
<tr>
<td>589</td>
<td><a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a></td>
</tr>
<tr>
<td>423</td>
<td><a href="mailto:domexuser2@gmail.com">domexuser2@gmail.com</a></td>
</tr>
<tr>
<td>347</td>
<td><a href="mailto:domexuser3@gmail.com">domexuser3@gmail.com</a></td>
</tr>
<tr>
<td>268</td>
<td><a href="mailto:ips@mail.ips.es">ips@mail.ips.es</a></td>
</tr>
<tr>
<td>252</td>
<td><a href="mailto:premium-server@thawte.com">premium-server@thawte.com</a></td>
</tr>
<tr>
<td>243</td>
<td><a href="mailto:someone@example.com">someone@example.com</a></td>
</tr>
<tr>
<td>243</td>
<td><a href="mailto:CPS-requests@verisign.com">CPS-requests@verisign.com</a></td>
</tr>
<tr>
<td>220</td>
<td><a href="mailto:domexuser2@live.com">domexuser2@live.com</a></td>
</tr>
<tr>
<td>194</td>
<td><a href="mailto:domexuser1@hotmail.com">domexuser1@hotmail.com</a></td>
</tr>
<tr>
<td>184</td>
<td><a href="mailto:domexuser1@live.com">domexuser1@live.com</a></td>
</tr>
</tbody>
</table>

sqlite>
```

Enable with command line option:

```
-S write_feature_sqlite3=YES
```
$ bulk_extractor -o output mydisk.raw
bulk_extractor is a command line tool.

$ ./bulk_extractor -o out-1 disk.img
bulk_extractor version: 1.5.0-alpha9
Hostname: mncrnpseu.local
Input file: disk.img
Output directory: out-1
Disk Size: 5164080
Threads: 24
Attempt to open bulk_extractor
All data are read; waiting for threads to finish...
Time elapsed waiting for 1 thread to finish:
  (timeout in 60 min.)
All Threads Finished!
Producer time spent waiting: 0 sec.
Average consumer time spent waiting: 1.96071 sec.
MD5 of Disk Image: d85cf891e6297e71c8d7ae4368bf5eb6
Phase 2. Shutting down scanners
Phase 3. Creating Histograms
Elapsed time: 2.05625 sec.
Total MB processed: 5
Overall performance: 2.51141 MBytes/sec (0.104642 MBytes/sec/thread)
Total email features found: 0
$
Help is always available:

$ src/bulk_extractor -h
bulk_extractor version 1.3b6 $Rev: 10046 $
Usage: src/bulk_extractor [options] imagefile
    runs bulk extractor and outputs to stdout a summary of what was found where

Required parameters:
    imagefile     - the file to extract
or  -R filedir  - recurse through a directory of files
    SUPPORT FOR E01 FILES COMPILLED IN
    SUPPORT FOR AFF FILES COMPILLED IN
    EXIV2 COMPILLED IN

    -o outdir     - specifies output directory. Must not exist.
    bulk_extractor creates this directory.

-h updates automatically depending on how bulk_extractor is compiled.
   - Disk image formats supported (E01, AFF)
   - Compiled-in scanners that are enabled
   - Plug-ins that are loaded at startup.
bulk_extractor input and output functions

Options change the behavior of the scanner:

Options:
- `-b banner.txt` - Add banner.txt contents to the top of every output file.
- `-r alert_list.txt` - a file containing the alert list of features to alert (can be a feature file or a list of globs) (can be repeated.)
- `-w stop_list.txt` - a file containing the stop list of features (white list (can be a feature file or a list of globs) (can be repeated.)
- `-F <rfile>` - Read a list of regular expressions from <rfile> to find
- `-f <regex>` - find occurrences of <regex>; may be repeated. results go into find.txt

- `-b` — Adds a “classification banner” to every output file.
- `-r`, `-w` — alert list (red list) and stop list (white list) features.
- `-F`, `-f` — Search for keywords in all data
Normally bulk_extractor will run one analysis thread per core. You can make it use less cores if you need to leave some free:

```
-j NN  - Number of analysis threads to run (default 8)
```

Output directory: regress-1.3b7-norm-01
Disk Size: 2106589184
Threads: 8
Phase 1.
  9:33:28 Offset 0MB (0.00%) Done in n/a at 09:33:27
  9:33:29 Offset 16MB (0.80%) Done in  0:00:58 at 09:34:27
  9:33:29 Offset 33MB (1.59%) Done in  0:00:48 at 09:34:17
  9:33:41 Offset 50MB (2.39%) Done in  0:08:27 at 09:42:08
  9:33:41 Offset 67MB (3.19%) Done in  0:06:42 at 09:40:23
There are many configurable options.

The -s option allows scanners to have settable tuning parameters:

- `S work_start_work_end=YES` Record work start and end of each scanner in report.xml file
- `S enable_histograms=YES` Disable generation of histograms
- `S hash_alg=md5` Specifies hash algorithm to be used for all hash calculations
- `S dup_data_alerts=NO` Notify when duplicate data is not processed
- `S write_feature_files=YES` Write features to flat files
- `S write_feature_sqlite3=NO` Write feature files to report.sqlite3
- `S report_read_errors=YES` Report read errors
- `S ssn_mode=0` 0=Normal; 1=No `SSN' required; 2=No dashes required (accts)
- `S min_phone_digits=6` Min. digits required in a phone (accts)
- `S carve_net_memory=NO` Carve network memory structures (net)

Use -h for a list of all options.
Individual scanners can be enabled or disabled.

These scanners disabled by default; enable with -e:

- `e base16` - enable scanner base16
- `e facebook` - enable scanner facebook
- `e outlook` - enable scanner outlook
- `e sceadan` - enable scanner sceadan
- `e wordlist` - enable scanner wordlist
- `e xor` - enable scanner xor

These scanners enabled by default; disable with -x:

- `x accts` - disable scanner accts
- `x aes` - disable scanner aes
- `x base64` - disable scanner base64
- `x elf` - disable scanner elf
- `x email` - disable scanner email
- `x exif` - disable scanner exif
- `x find` - disable scanner find
- `x gps` - disable scanner gps
- `x gzip` - disable scanner gzip
- `x hiber` - disable scanner hiber
- `x httplogs` - disable scanner httplogs
- `x json` - disable scanner json
- `x kml` - disable scanner kml
- `x net` - disable scanner net
- `x pdf` - disable scanner pdf
- `x rar` - disable scanner rar
- `x sqlite` - disable scanner sqlite
- `x vcard` - disable scanner vcard
- `x windirs` - disable scanner windirs
- `x winlnk` - disable scanner winlnk
- `x winpe` - disable scanner winpe
- `x winprefetch` - disable scanner winprefetch
- `x zip` - disable scanner zip

Don’t assume that you should enable every scanner!

Discussed in next sections
<table>
<thead>
<tr>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>476</td>
<td>Jul 7 23:50</td>
<td>aes_keys.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>alerts.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>2743</td>
<td>Jul 7 23:59</td>
<td>ccn.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>454</td>
<td>Jul 8 00:03</td>
<td>ccn_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>ccn_track2.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8 00:03</td>
<td>ccn_track2_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>23369167</td>
<td>Jul 8 00:03</td>
<td>domain.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>185266</td>
<td>Jul 8 00:03</td>
<td>domain_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>elf.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>1719842</td>
<td>Jul 8 00:03</td>
<td>ether.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>35073</td>
<td>Jul 8 00:03</td>
<td>email_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>23961</td>
<td>Jul 8 00:00</td>
<td>ether.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>337</td>
<td>Jul 8 00:03</td>
<td>email_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>11188830</td>
<td>Jul 8 00:03</td>
<td>exif.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>find.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>1112</td>
<td>Jul 8 00:01</td>
<td>gps.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>hex.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>95835</td>
<td>Jul 8 00:03</td>
<td>ip.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>11603</td>
<td>Jul 8 00:03</td>
<td>ip_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>2025702</td>
<td>Jul 8 00:03</td>
<td>json.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>kml.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>194991</td>
<td>Jul 8 00:03</td>
<td>packets.pcap</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>21343</td>
<td>Jul 8 00:03</td>
<td>report.xml</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>3782598</td>
<td>Jul 8 00:03</td>
<td>rfc822.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>213746</td>
<td>Jul 8 00:03</td>
<td>tcp.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>61255</td>
<td>Jul 8 00:03</td>
<td>tcp_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>59469</td>
<td>Jul 8 00:03</td>
<td>telephone.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>6612</td>
<td>Jul 8 00:03</td>
<td>telephone_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>67205326</td>
<td>Jul 8 00:03</td>
<td>url.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8 00:03</td>
<td>url_facebook-id.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>5706665</td>
<td>Jul 8 00:03</td>
<td>url_histogram.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8 00:03</td>
<td>url_microsoft-live.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>8504</td>
<td>Jul 8 00:03</td>
<td>url_searches.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>151673</td>
<td>Jul 8 00:03</td>
<td>url_services.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7 23:48</td>
<td>vcard.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>18549729</td>
<td>Jul 8 00:03</td>
<td>windirs.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>29051041</td>
<td>Jul 8 00:03</td>
<td>winpe.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>1984759</td>
<td>Jul 8 00:03</td>
<td>winprefetch.txt</td>
</tr>
<tr>
<td>-rw-</td>
<td>simsong</td>
<td>staff</td>
<td>34128889</td>
<td>Jul 8 00:03</td>
<td>zip.txt</td>
</tr>
</tbody>
</table>
Optimistic Decoding scanners

- **base64** — Decodes Base64 encoded data (email attachments & SSL certificates)
- **gzip** — Browser cache
- **hiberfile** — Decompresses fragments of Windows hibernation files.
- **pdf** — Extracts text from PDF files
- **rar** — Decompressed unencrypted RAR files; carves encrypted RAR files
- **zip** — decompresses ZIP’ed files, optionally carves ZIP components.
- **base16** (disabled by default*) — finds HEXADECIMAL strings and decodes them
- **outlook*** (disabled by default*) — Outlook Compressible Encryption; obsolete
- **xor** (disabled by default) — applies any XOR mask to all input data; good for malware.

Identity scanners:

- **accts** — Credit Card numbers, Track 2 data, Telephone Numbers, and “PII”
  — **DOB; SSNs; TeamViewer***; etc.
- **email** — email addresses, IP addresses, Email message headers, MAC addresses, URLs
- **gps** — Looks for Garmin-formatted trackpoint XML
More scanners in bulk_extractor 1.5

Information on Linux and Windows operating system structures
- **elf** — Linux Executables
- **windirs** — Windows directory entries (NTFS and FAT32)
- **winpe** — Windows executables
- **winprefetch** — Windows prefetch files
- **winlnk*** — Windows LNK files

Web-related scanners:
- **json** — Properly-formatted JSON
- **kml** — KML files
- **facebook*** — Facebook HTML
- **httplogs*** — Looks for fragments of HTTP log files
- **vcard** — VCARD entries
More scanners in bulk_extractor 1.5

Misc:
- **aes** — Searches for scheduled AES keys in RAM dumps
- **wordlist** — Simple wordlist generator (la-strings is better)
- **net** — IP packets & TCP/IP memory structures
- **find** — Simple word search
- **sqlite** — identifies SQLite3 databases and optionally carves them.

Plug-ins for other NPS projects:
- **hashdb** — Builds or Searches an NPS “sector hash database.”
- **sceadan** — Runs UTSA file type identification system
Why use bulk_extractor to find Windows executables?

BE works with incomplete executables:
- Only needs the first 4K
- Produces MD5 hash of first 4K (distinct for most executables)
- Decodes all available PE header fields

BE finds executables other approaches miss:
- Orphans not in the file system
- Fragments in RAM
- BASE64 encoded PE files (email attachments)
- GZIP-encoded PE files (HTTP downloads)
- RAR-encoded PE files
Why use bulk_extractor to find Windows LNK files?

Windows LNK files provide information about:

- Most Recently Used files
- Shortcuts
- Other system information.

LNK information is provided as DFXML data in the feature file:

```
263671296        E:\x5Cz-netvampire33_2\x5C10.jpg
<lnk><atime>2003-08-09T16:00:00Z</atime><ctime>2000-10-25T14:49:40Z</ctime><path>E:\x5Cz-netvampire33_2\x5C10.jpg</path><wtime>1999-06-02T23:46:08Z</wtime></lnk>
```

- Reformats as:

```
263671296        E:\x5Cz-netvampire33_2\x5C10.jpg
<lnk>  <atime>2003-08-09T16:00:00Z</atime>
<ctime>2000-10-25T14:49:40Z</ctime>
<path>E:\x5Cz-netvampire33_2\x5C10.jpg</path>
<wtime>1999-06-02T23:46:08Z</wtime> </lnk>
```

BE will find LNK files that were deleted and “lost”
Why use bulk_extractor to find Prefetch files?

Windows prefetch files contain:
- Name of executable
- Time last run
- Total Runs
- Linked DLLs

Windows deletes prefetch files when there are more than 129. bulk_extractor finds all of the prefetch files on the drive and decodes the contents as XML:

```
13688320    SHMGRATE.EXE    <prefetch><os>Windows XP</os><filename>SHMGRATE.EXE<filename><header_size>152<header_size><atime>2005-08-08T14:42:40Z</atime><runs>1</runs><filenames>...
```

Drive 0844.aff has 400+ prefetch files on the drive. Use them to:
- Infer when a program was last run.
- Gauge a program’s popularity
Why use bulk_extractor to find RAR files?

rar.txt — A large file says that there were lots of RAR files

- rar.txt will give you the file names, flags, file size, and other data:


You can scan for encrypted files.

<encrypted>true</encrypted>

You can carve objects from within RAR files

- JPEGs, PDFs, etc.
Why use bulk_extractor for web server logs?

Because they are sometimes present! Unknowingly! e.g. IN10-0512:

```
4406559744  GET  /n/p?module=8814&error=0&language=09.01&product=Norton %20AntiVirus&version=16.0.0.125&hbguid=839fa0aa-d7bf-11df-bb3c-001cc0560f34&c=1425011&psn=CHT8VVM9GBY&m=14483463&b=200&a=0&h=351 HTTP/1.1\x0D
GET /n/p?module=8814&error=0&language=09.01&product=Norton %20AntiVirus&version=16.0.0.125&hbguid=839fa0aa-d7bf-11df-bb3c-001cc0560f34&c=1425011&psn=CHT8VVM9GBY&m=14483463&b=200&a=0&h=351 HTTP/1.1\x0D
```

Found on the NPS 2TB disk:

```

```
bulk_extractor 1.5 will carve encoded files. These are missed by other scanners.

Some scanners can “carve.” Each carving scanner has a carving mode.

- Carving mode 0 — carve nothing
- Carving mode 1 — Only carve if file was “encoded.”
  — e.g. a JPEG won’t be carved, but a BASE64-encoded JPEG will be carved.
  — These are the files missed by conventional carvers.
- Carving mode 2 — Carve everything.

Carved files are binned in the output directory:

```
$ ls -l out-TH0001_0028/jpeg_carved/000/
  2905 Jul  2 06:16 10410502381-ZIP-29.jpg
  15452 Jul  2 06:16 10602684084-BASE64-0.jpg
  1395 Jul  2 06:18 11547112759-ZIP-2184645-ZIP-1320-ZIP-0.jpg
```

Carving scanners:

- exif — JPEGs
- kml — KML files
- net — IP packets
- RAR — RAR-compressed files

sqlite - Sqlite3 databases (only contiguous)
vcard — Contacts (VCARDS)
Bugfix #1: Fix to scan_base64 scanner

Bulk_extractor 1.4 scan_base64 missed many BASE64 regions.
- Fixed in BE1.5!

<table>
<thead>
<tr>
<th>Disk Image</th>
<th>BE14 base64 emails</th>
<th>BE15 base64 emails</th>
</tr>
</thead>
<tbody>
<tr>
<td>domexusers</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>NPS 2TB Drive</td>
<td>1</td>
<td>4,760</td>
</tr>
</tbody>
</table>

Most BASE64-encoded email addresses are from:
- SSL certificates
- email attachments
Bugfix #2:
Improved stoplist performance

bulk_extractor supports two kinds of “stop lists” (black lists)

- Word-based:
  0%4@2gy2kj.es
  0%f@m4.bh
  0%p@vcp.sr
  0%s@earthlink.net
  0+anwwhtbp8fa@hpqeqa

- Context-based:
  11996200  CPS-requests@verisign.com  m; by E-mail at CPS-requests@verisign.com; or\x0Aby mail at
  1247926963  Thierry.Mayeur@lotus.com  ing addresses:\x0D\x0DThierry.Mayeur@lotus.com\x0DTMayeur@ca.ibm.
  124792718  TMayeur@ca.ibm.com ayeur@lotus.com\x0DTMayeur@ca.ibm.com\x0D\x0D2.
What's new\x0D
  212163670  CPS-requests@verisign.com  m; by E-mail at CPS-requests@verisign.com; or\x0Aby mail at
  352717318  firstname_lastname@company.com  main, such as:
    'firstname_lastname@company.com(foobar)@Notes_d

Context-based stop-lists are feature files concatenated together.
BE1.5 stoplist:
NIST ran bulk_extractor over the entire NSRL
Each NSRL output is a ZIP file — a bulk_extractor report

```
$ unzip -l /corp/nus/bulk_output/nsrl/43-1.Metro-X\ X11.6.3.1.2.Metro\ Link\ Inc..NOCARVE.zip
```

<table>
<thead>
<tr>
<th>Length</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./aes_keys.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./alerts.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./ccn.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./ccn_histogram.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./ccn_track2.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./elf.txt</td>
</tr>
<tr>
<td>484</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./ether.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./exif.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./find.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./find_histogram.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./gps.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc/httplogs.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./jp.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./jp_histogram.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./jpeg_carved.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc./kml.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc/rar.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc/sqlite.txt</td>
</tr>
<tr>
<td>0</td>
<td>07-05-2014</td>
<td>21:15</td>
<td>43-1.Metro-X X11.6.3.1.2.Metro Link Inc/unrar_carved.txt</td>
</tr>
</tbody>
</table>
# BULK_EXTRACTOR-Version: 1.5.0-alpha2 ($Rev: 10844 $)
# Feature-Recorder: email
# Filename: /Volumes/SanVol1/iRepository/media/43-1/ff81e9459a5fa057c2922ba0656414e336635248/ff81e9459a5fa057c2922ba0656414e336635248.img
# Histogram-File-Version: 1.1
n=101 cps-requests@verisign.com 
n=8  msccatus@microsoft.com 
n=6  expresstools@autodesk.com 
n=3  anonymous@microsoft.com 
n=3  cindyl@neucom.com (utf16=3) 
n=3  zeusprod@aol.com (utf16=3) 
n=2  aaron.ryan@prd-foods.com 
n=2  aj.stensen@prd-foods.com 
n=2  alfonso.camacho@prd-foods.com 
n=2  bill.hughes@prd-foods.com 
n=2  bill.mcgregor@prd-foods.com 
n=2  bruce.gillmore@prd-foods.com 
n=2  bryan.partridge@prd-foods.com 
n=2  carlos.perez@prd-foods.com 
n=2  carroll.engdahl@prd-foods.com 
n=2  chandra.singh@prd-foods.com 
n=2  chela.james@prd-foods.com 
n=2  christian.ludwig@prd-foods.com 
n=2  claytonc@neucom.com (utf16=2) 
n=2  cliffordy@neucom.com (utf16=2) 
n=2  craig.anderson@prd-foods.com
In BE1.4, all words with “.” were treated as regular expressions.
- Stoplists of email addresses were too slow to be usable.

**BE1.5 performance scanning domexusers (40GB test image)**

<table>
<thead>
<tr>
<th>Stop list</th>
<th>Lines</th>
<th>Time to process</th>
<th>Found email</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>304 seconds</td>
<td>1050 distinct, 8744 total</td>
</tr>
<tr>
<td>NSRL email words</td>
<td>502,697 (12MB)</td>
<td>296 seconds</td>
<td>286 distinct, 3822 total</td>
</tr>
<tr>
<td>NSRL email context</td>
<td>45,231,320 (4.4GB)</td>
<td>371 seconds</td>
<td>364 distinct, 3737 total</td>
</tr>
</tbody>
</table>
Example output with and without stoplist

Baseline:
n=609 domexuser1@gmail.com (utf16=303)
n=455 domexuser2@gmail.com (utf16=225)
n=359 domexuser3@gmail.com (utf16=204)
n=268 ips@mail.ips.es
n=252 premium-server@thawte.com
n=243 cps-requests@verisign.com (utf16=3)
n=243 someone@example.com (utf16=234)
n=221 domexuser2@live.com (utf16=59)
n=198 domexuser1@hotmail.com (utf16=80)
n=185 domexuser1@live.com (utf16=59)
n=175 domexuser2@hotmail.com (utf16=97)
n=145 inet@microsoft.com
n=115 example@passport.com (utf16=115)
n=115 myname@msn.com (utf16=115)
n=94 info@valicert.com
n=91 piracy@microsoft.com (utf16=91)
n=80 certificate@trustcenter.de
n=78 name_123@hotmail.com (utf16=78)
n=74 talkback@mozilla.org (utf16=12)
n=69 hewitt@netscape.com (utf16=1)
n=64 lord@netscape.com
n=53 someone@microsoft.com (utf16=48)
n=51 mcgreer@netscape.com
n=48 domexuser1%40gmail.com@imap.gmail.com (utf16=16)
n=43 49091023.6070302@gmail.com (utf16=22)
n=43 73a94919-ff6b-4e3f-938e-fb39bbc7497c@gmail.com (utf16=43)
n=43 9name_123@hotmail.com (utf16=43)
n=42 domex2@rad.li (utf16=42)
n=36 49091664.70508@gmail.com (utf16=22)
n=35 domex1@rad.ms (utf16=35)
n=33 domexuser2@gmail.com (utf16=8)
n=27 domex1@www.ms (utf16=27)
n=26 premium-server@thawte.com
n=25 outldomexuser2@gmail.com-00000002.ps (utf16=25)
n=23 2domexuser2@gmail.com (utf16=23)
n=23 314d3a220810291941w4b52597fh206faba1e5063365@mail.gmail.com (utf16=23)
n=23 domex2@adopt.eu (utf16=23)
n=20 domex2@bl135w.blu135.mail.li (utf16=20)
n=19 alguem@exemplo.pt (utf16=19)
n=19 cph@99841.pa
n=18 outldomexuser2@gmail.com-00000002.pst.tm (utf16=18)
managelinks.aspx%3fmkt%3den-us%26noteid%3dnote.linked%26notelevel%3d1%26notesec%3d0%26username%3ddomexuser1@

With Stoplist:
n=609 domexuser1@gmail.com (utf16=303)
n=455 domexuser2@gmail.com (utf16=225)
n=359 domexuser3@gmail.com (utf16=204)
n=221 domexuser2@live.com (utf16=59)
n=198 domexuser1@hotmail.com (utf16=80)
n=185 domexuser1@live.com (utf16=59)
n=175 domexuser2@hotmail.com (utf16=97)
n=67 talkback@mozilla.org (utf16=6)
n=48 domexuser1%40gmail.com@imap.gmail.com (utf16=16)
n=43 49091023.6070302@gmail.com (utf16=22)
n=43 73a94919-ff6b-4e3f-938e-fb39bbc7497c@gmail.com (utf16=43)
n=43 9name_123@hotmail.com (utf16=43)
n=42 domex2@rad.li (utf16=42)
n=36 49091664.70508@gmail.com (utf16=22)
n=35 domex1@rad.ms (utf16=35)
n=33 domexuser2@gmail.com (utf16=8)
n=27 domex1@www.ms (utf16=27)
n=26 premium-server@thawte.com
n=25 outldomexuser2@gmail.com-00000002.ps (utf16=25)
n=23 2domexuser2@gmail.com (utf16=23)
n=23 314d3a220810291941w4b52597fh206faba1e5063365@mail.gmail.com (utf16=23)
n=23 domex2@adopt.eu (utf16=23)
n=20 domex2@bl135w.blu135.mail.li (utf16=20)
n=19 alguem@exemplo.pt (utf16=19)
n=19 cph@99841.pa
n=18 outldomexuser2@gmail.com-00000002.pst.tm (utf16=18)
managelinks.aspx%3fmkt%3den-us%26noteid%3dnote.linked%26notelevel%3d1%26notesec%3d0%26username%3ddomexuser1@
The included program bulk_diff.py will compare the two reports

$ python bulk_diff.py --help
Usage: usage: bulk_diff.py [options] <pre> <post>
<pre> and <post> may be a bulk_extractor output directory or a zip file of a report.

Options:
- h, --help            show this help message and exit
--smaller             Also show values that didn't change or got smaller
--same               Also show values that didn't change
--tabdel=TABLE       Specify a tab-delimited output file for easy import into Excel
--html=HTML           HTML output. Argument is file name base

Usage:
$ python bulk_diff.py out-domexusers-stop-context out-domexusers-baseline
Files only in out-domexusers-stop-context:
  ip_stopped.txt (0 lines)
  email_stopped.txt (5007 lines)
  rar_stopped.txt (0 lines)
  rfc822_stopped.txt (0 lines)
  pii_stopped.txt (0 lines)
  ...

...
The differences are due to email addresses in distribution software being suppressed.

ccn_histogram.txt: No differences

ccn_track2_histogram.txt: No differences

domain_histogram.txt: No differences

e-mail_histogram.txt:

<table>
<thead>
<tr>
<th># in PRE</th>
<th># in POST</th>
<th>∆</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>268</td>
<td>268</td>
<td><a href="mailto:ips@mail.ips.es">ips@mail.ips.es</a></td>
</tr>
<tr>
<td>0</td>
<td>243</td>
<td>243</td>
<td><a href="mailto:cps-requests@verisign.com">cps-requests@verisign.com</a></td>
</tr>
<tr>
<td>7</td>
<td>243</td>
<td>236</td>
<td><a href="mailto:someone@example.com">someone@example.com</a></td>
</tr>
<tr>
<td>26</td>
<td>252</td>
<td>226</td>
<td><a href="mailto:premium-server@thawte.com">premium-server@thawte.com</a></td>
</tr>
<tr>
<td>0</td>
<td>145</td>
<td>145</td>
<td><a href="mailto:inet@microsoft.com">inet@microsoft.com</a></td>
</tr>
<tr>
<td>0</td>
<td>115</td>
<td>115</td>
<td><a href="mailto:example@passport.com">example@passport.com</a></td>
</tr>
<tr>
<td>0</td>
<td>115</td>
<td>115</td>
<td><a href="mailto:myname@msn.com">myname@msn.com</a></td>
</tr>
<tr>
<td>0</td>
<td>91</td>
<td>91</td>
<td><a href="mailto:piracy@microsoft.com">piracy@microsoft.com</a></td>
</tr>
<tr>
<td>8</td>
<td>94</td>
<td>86</td>
<td><a href="mailto:info@valicert.com">info@valicert.com</a></td>
</tr>
<tr>
<td>0</td>
<td>80</td>
<td>80</td>
<td><a href="mailto:certificate@trustcenter.de">certificate@trustcenter.de</a></td>
</tr>
<tr>
<td>0</td>
<td>78</td>
<td>78</td>
<td><a href="mailto:name_123@hotmail.com">name_123@hotmail.com</a></td>
</tr>
<tr>
<td>0</td>
<td>69</td>
<td>69</td>
<td><a href="mailto:hewitt@netscape.com">hewitt@netscape.com</a></td>
</tr>
<tr>
<td>0</td>
<td>64</td>
<td>64</td>
<td><a href="mailto:lord@netscape.com">lord@netscape.com</a></td>
</tr>
<tr>
<td>0</td>
<td>51</td>
<td>51</td>
<td><a href="mailto:mcgreer@netscape.com">mcgreer@netscape.com</a></td>
</tr>
<tr>
<td>4</td>
<td>53</td>
<td>49</td>
<td><a href="mailto:someone@microsoft.com">someone@microsoft.com</a></td>
</tr>
<tr>
<td>0</td>
<td>47</td>
<td>47</td>
<td><a href="mailto:neil@parkwaycc.co.uk">neil@parkwaycc.co.uk</a></td>
</tr>
<tr>
<td>3</td>
<td>43</td>
<td>40</td>
<td><a href="mailto:mazrob@panix.com">mazrob@panix.com</a></td>
</tr>
<tr>
<td>0</td>
<td>37</td>
<td>37</td>
<td><a href="mailto:server-certs@thawte.com">server-certs@thawte.com</a></td>
</tr>
</tbody>
</table>
bulk_extractor 1.5
scanners and output
There are four main categories of feature files:

Identity Information:
- Domain Names; Email addresses; URLs
- Search terms; Facebook IDs; JSON data
- KML files
- VCARDS
- find output

Technical Info:
- ZIP files; EXIF data

Network Information:
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.

Information about executables:
- ELF & PE headers; Windows Prefetch files
There are four main categories of feature files:

**Identity Information:**
- Domain Names; Email addresses; URLs
- Search terms; Facebook IDs; JSON data
- KML files
- VCARDS
- find output

**Technical Info:**
- ZIP files; EXIF data

**Network Information:**
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.

**Information about executables:**
- ELF & PE headers; Windows Prefetch files
ccn.txt — potential credit card numbers.

bulk_extractor’s credit card number finder:
- Considers pattern of digits; Luhn algorithm; distribution of digits; local context
- Frequently alerts on “false positives,” so be careful!

In this example:
- 5273347458642687 looks like a valid CCN from the context (\x0A is a new line)
- 4015751530102097 looks like a random number in a piece of JavaScript
  — Notice it was compressed! offset 4814857216 starts a GZIP stream; +793 bytes is CCN
- “Inexact Rounde” is actually from the Python source code
  — http://svn.python.org/projects/python/branches/pep-0384/Lib/test/decimaltestdata/ddAdd.decTest
ccn_histogram.txt: a histogram of the potential credit card numbers

Normally this is a great way to find the real numbers...

| n=20 | 6543210123456788 |
| n=2  | 4015751530102097 |
| n=2  | 4920919202474441 |
| n=1  | 4857994530998756 |
| n=1  | 4909616081396134 |
| n=1  | 5235714985079914 |
| n=1  | 5273347458662687 |
| n=1  | 5578481572827551 |
| n=1  | 5678901234560000 |
| n=1  | 5700122152274696 |

This time it’s a great way to find that python test data!
ccn_track2.txt contains potential “track 2” credit card number information

<table>
<thead>
<tr>
<th>Length</th>
<th>Date</th>
<th>Time</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>476</td>
<td>8-Jul-2012</td>
<td>01:50:32</td>
<td>charlie-2009-12-11/aes_keys.txt</td>
</tr>
<tr>
<td>0</td>
<td>8-Jul-2012</td>
<td>01:48:36</td>
<td>charlie-2009-12-11/alerts.txt</td>
</tr>
<tr>
<td>2743</td>
<td>8-Jul-2012</td>
<td>01:59:24</td>
<td>charlie-2009-12-11/ccn.txt</td>
</tr>
<tr>
<td>454</td>
<td>8-Jul-2012</td>
<td>02:03:14</td>
<td>charlie-2009-12-11/ccn_histogram.txt</td>
</tr>
<tr>
<td>0</td>
<td>8-Jul-2012</td>
<td>01:48:36</td>
<td>charlie-2009-12-11/ccn_track2.txt</td>
</tr>
<tr>
<td>0</td>
<td>8-Jul-2012</td>
<td>02:03:14</td>
<td>charlie-2009-12-11/ccn_track2_histogram.txt</td>
</tr>
</tbody>
</table>

... In this case we don’t have any track 2 data...
domain.txt is a list of all the potential “domains” and host names that were found. Sources include potential URLs, email, dotted quads.


xOD\xA\x0D\xA

169692672-GZIP-4139  us.ard.yahoo.com  8" href="http://us.ard.yahoo.com/
SIG=15s920d26/M

148770304-GZIP-63217  www.bakersfield.com  n value="http://
www.bakersfield.com">CA, Bakersfiel

148770304-GZIP-63295  www.thebakersfieldchannel.com  n value="http://
www.thebakersfieldchannel.com">CA, Bakersfiel

27766700  205.155.65.61  ustang.nps.edu [205.155.65.61]\xCD\xA\x09(using
27766902  m57.biz \xCD\xA\x09for <charlie@m57.biz>; Mon, 16 Nov 2

Note:
- UTF-16 is “escaped” as Python-style — \x00h\x00o\x00t means “hot”
- Domains are common in compressed data
domain_histogram.txt is a histogram of the domains...

```
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# bulk_extractor-Version: 1.3b1-dev2
# Filename: /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01
# Feature-Recorder: domain
# Histogram-File-Version: 1.1
n=10749  www.w3.org
n=6670   chroniclingamerica.loc.gov
n=6384   openoffice.org
n=5998   www.uspto.gov
n=5733   www.mozilla.org
n=5212   www.osti.gov
n=4952   www.microsoft.com
n=4474   patft.uspto.gov
n=4468   www.gpo.gov
n=3653   www.verisign.com
n=3167   www.google.com
n=3150   www.wipo.int
n=2733   news.bbc.co.uk
n=2595   crl.microsoft.com
```

Many of these domains are part of the operating system. Some aren’t.
email.txt is similar to domain.txt, but has the potential email addresses!

50395384 n\000\00m\00b\00r\00e\00\_\001\002\003\00@\00h\00o
\00t\00m\00a\00i\001\000.\00c\00o\00m\00 e\00m\00p\001\000
\00: \00\00A\00\09\00n\00o\00m\00b\00r\00e\00\_\001\002\003\00@\00h
\00o\00t\00m\00a\00i\001\000.\00c\00o\00m\00m\00A\00\09\00m\00i\00n
\00o\00m\00b\00

50395432 m\00i\00m\00b\00r\00e\00 \00m\00n\00o\00m\00b\00r\00e\00\_\002\003\00\_\00h
\00m\00b\00r\00e\00\_\001\002\003\00\_\00n
\00m\00h\00i\00m\00b\00r\00e\00\_\001\002\003\00\_\00n
\00m\00h\00i\00m\00b\00r\00e\00\_\001\002\003\00\_\00n

— minombre@msn.com — myname@msn.com?
— 50395384 is very early in the disk...

Further down we see:

828564544 charlie@m57.biz (37190)\0D\0A\09 for <charlie@m57.biz>; Fri, 20 Nov 2

828564992 4B01C378.3060603@m57.biz 0\0D\0ARefers:
<4B01C378.3060603@m57.biz>\0D\0ATo: charlie@m

828565023 charlie@m57.biz 3@m57.biz>\0D\0ATo: charlie@m57.biz\0D
\0ASubject: Still
email_histogram.txt shows a histogram of all potential email addresses

Clearly the histogram makes a difference:

- Charlie’s email is #2 (it would probably be #1 if the disk had been used for more than 3 weeks)
- Charlie’s email appeared 651 times; 120 of those were in UTF-16.
- Many of these email addresses are from the software (ajbanck@planet.nl is in Mozilla Calendar)
find.txt is the result of the ‘find’ command

```
-rw-r--r--@ 1 simsong staff 0 Jul 7 23:48 find.txt
```

But we can run with the find command (-f) to do a string search.
- Here we look for any mentions of ‘nps.edu’ (any case) in charlie-2009-12-11

```
$ bulk_extractor -f '[nN][pP][sS].[eE][dD][uU]' -o charlie-2009-12-11-find /corp/nps/scenarios/2009-m57-patents/drives-redacted/charlie-2009-12-11.E01
...
elapsed time: 1787.12 seconds
$
- The string search is executed as a first-class scanner (so it goes in compressed data)

```
 27766691  nps.edu  ps.edu  (mustang.nps.edu [205.155.65.61]
 27767031  nps.edu  http://mustang.nps.edu:80/cgi-bin/mark
...
3449724105  nps.edu  wall at mustang.nps.edu. I'm sorry to i
3445904906  nps.edu  ED0A1F1@mustang.nps.edu)(25C=fe0)(261\x0D\x0A ...
9976666871  nps.edu  $\xA0"\x1B cifs/domex.nps.edu@DOMEX.NPS.EDU\x00\x00
9976666885  NPS.EDU  x.nps.edu@DOMEX.NPS.EDU\x00\x00\x00\x0D\x00\x01\x00Fil\xE5\x01\x15\x02
```

Note that these “domains” are not included in the domain histogram!
Provides offset, JSON, and MD5 of JSON

- Use the MD5 for deduplication

```
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# bulk_extractor-Version: 1.3b1-dev2
# Filename: /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01
# Feature-Recorder: json
# Feature-File-Version: 1.1

5091418457 [6, 4, 6, 4] 7ea5995a7acbd301b98e15b50b723e2b
5091418525 [6, 4, 6, 4] 7ea5995a7acbd301b98e15b50b723e2b

10002203123 {"url":"http://patft.uspto.gov/netacgi/nph-Parser?
Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetacgi%2FPTO%2Fsearch-
bool.html&r=26265.169749810822&f=G&l=50&col=AND&d=PTXT&s1=mortality&OS=mortality&
RS=mortality","title":"United States Patent: 4035489","ID":63,"scroll":"0,0"}
81e95912dbb0e7e0966a9becf1c9f74a
```

Excellent for web pages:

- bulk_extractor is great at finding JSON in compressed streams, HIBER files, etc.
- There is a huge amount of stuff here
We would like to have better reporting of mail headers.

—Combining email address and name
Beware — many are tech support!

<table>
<thead>
<tr>
<th>Phone</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>88850883</td>
<td>(800) 563-9048</td>
<td>Information Centre: (800) 563-9048</td>
</tr>
<tr>
<td>88850995</td>
<td>(905) 568-4494</td>
<td>Microsoft Windows (905) 568-4494</td>
</tr>
<tr>
<td>88851056</td>
<td>(905) 568-2294</td>
<td>Other Microsoft Ice Components: (905) 568-2294</td>
</tr>
<tr>
<td>88851111</td>
<td>(905) 568-3503</td>
<td>Priority Technical Support: (905) 568-3503</td>
</tr>
<tr>
<td>88851162</td>
<td>(800) 668-7975</td>
<td>Text Telephone: (800) 668-7975</td>
</tr>
<tr>
<td>88851208</td>
<td>(905) 568-9641</td>
<td>Phone (TTY/TDD): (905) 568-9641</td>
</tr>
<tr>
<td>88851367</td>
<td>(809) 273-3600</td>
<td>Phone: (809) 273-3600</td>
</tr>
</tbody>
</table>

Some are bogus:

<table>
<thead>
<tr>
<th>Phone</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1003801194</td>
<td>252.227-7013</td>
<td>clause at DFARS 252.227-7013 or subparagraph</td>
</tr>
<tr>
<td>10051721420</td>
<td>118/150/1746</td>
<td>/filestorage/78/118/150/1746/1159/1268/Augus</td>
</tr>
</tbody>
</table>

And some are clearly legit:

<table>
<thead>
<tr>
<th>Phone</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6561037824-GZIP-28322</td>
<td>(831) 373-5555</td>
<td>Monterey - (831) 373-5555</td>
</tr>
<tr>
<td>6561037824-GZIP-29518</td>
<td>(831) 899-8300</td>
<td>Seaside - (831) 899-8300</td>
</tr>
<tr>
<td>6561037824-GZIP-31176</td>
<td>(831) 899-8300</td>
<td>Seaside - (831) 899-8300</td>
</tr>
</tbody>
</table>
Usually a better place to look for potential phone numbers

```
# Filename: /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01
# Feature-Recorder: telephone
# Histogram-File-Version: 1.1
n=42  4159618830
n=35  8477180400
n=24  2225552222
n=24  +27112570000
n=18  8005043248
n=15  2225551111
n=12  8772768437
n=11  2522277013
n=11  8662347350
n=9   1115554444
n=9   1771881984
n=8   4253532287
```

In version 1.3, non-digits are extracted from phone number.
Note that UTF-16 data is escaped:

```
%2Fpatft.uspto.gov%2Fnetacgi%2Fnph-Parser%3FSect1%3DPTO2%2526Sect2%3DHIT center>
\x0A<a href=http://patimg2.uspto.gov/.piw?Docid=07626151&homeurl=http%3A%2F
%2Fpatft.uspto.gov%2Fnetacgi%2Fnph-Parser%3FSect1%3DPTO2%2526Sect2%3DHITO
OFF%2526p%3D1%25

34913630 H\x00T\x00T\x00P\x00:\x00/\x00/\x00 n\x00a\x00s\x00 \x00W
\x00e\x00b\x00 \x00H\x00T\x00T\x00P\x00:\x00/\x00/\x00 \x00e\x00 \x00H\x00T\x00T
\x00P\x00S\x00

34913650 H\x00T\x00T\x00P\x00S\x00:\x00/\x00/\x00.\x00 T\x00P
\x00:\x00/\x00/\x00 \x00e\x00 \x00H\x00T\x00T\x00P\x00S\x00:\x00/\x00/\x00.\x00
\x00\x00N\x00o\x00 \x00e\x00n\x00t\x00a\x00

53952231 http://appft1.uspto.gov/netacgi/nph-Parser?
TERM1=20020186464&Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetacgi%2FPTO
%2Fsrchnum.html&r=0& =left><\x0A<a href="http://appft1.uspto.gov/netacgi/nph-
Parser?TERM1=20020186464&Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetacgi%2FPTO
%2Fsrchnum.html&r=0&f=S&l=50" target
```
url_histogram.txt: potential URLs from the disk
UTF-16 is converted to UTF8

n=2  http://ebiz1.uspto.gov/vision-service/ShoppingCart_P/AddToShoppingCart?
docNumber=7626465&backUrl1=http%3A//patft1.uspto.gov/netacgi/nph
n=2  http://ebiz1.uspto.gov/vision-service/ShoppingCart_P/AddToShoppingCart?
docNumber=7627056&backUrl1=http%3A//patft1.uspto.gov/netacgi/nph

Note:

n=1022  http://www.uspto.gov/patft/help/help.htm  (utf16=3)
n=992  http://www.uspto.gov/patft/index.html  (utf16=4)

Not all URLs are accurate:

n=3922  http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul  (utf16=2609)
n=859  http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul  (utf16=858)
url_facebok, url_histogram, url_microsoft-live, url_searches and url_services pull info out of URLs...

The most useful is url_searches.txt:

n=59    1
n=53    exotic+car+dealer
n=41    ford+car+dealer
n=34    2009+Shelby
n=25    steganography
n=23    General+Electric
n=23    time+travel
n=19    steganography+tool+free
n=19    vacation+packages
n=16    firefox
n=16    quicktime
n=14    7zip
n=14    fox+news
n=13    hex+editor

Searches frequently convey intent.
There are four main categories of feature files:

Identity Information:
- Domain Names; Email addresses; URLs
- Search terms; Facebook IDs; JSON data
- KML files
- VCARDs
- find output

Technical Info:
- ZIP files; EXIF data

Network Information:
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.

Information about executables:
- ELF & PE headers; Windows Prefetch files
aes_keys.txt — scheduled potential AES encryption keys, usually found in RAM, Swap, or hibernation files

- Three AES keys, appearing 1, 2 and 10 times
- Keys “01 02 03 04 05 06 07 … “ is a test vector seen on many Windows systems.
exif.txt is a list of every potential EXIF that is found on the drive

This feature file has a different internal formatting:

<table>
<thead>
<tr>
<th>offset</th>
<th>MD5 of first 4K of JPEG</th>
<th>XML encoding of EXIF</th>
</tr>
</thead>
</table>

These files are really hard to understand...

Fortunately, we have a program that turns it into a spreadsheet...
```bash
$ python bulk_extractor-1.3/python/post_process_exif.py exif.txt exif.csv
Input file: exif.txt
Output file: exif.csv
Scanning for EXIF tags...
There are 856 exif tags
$
$ open exif.csv
```

- Still not great, but at least you can search it and re-arrange the columns.
gps.txt shows times and potential GPS info extracted from JPEGs and Garmin XML files.

This is interesting because it’s data from other devices (cameras, etc.)
hex.txt is extracted hexadecimal strings of special lengths.

This disk image doesn’t have any...

Uses:

- emailed strings of MD5 codes, AES keys, etc.
- Anything else?
You will find most of the disk entries:

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Object Source</th>
<th>Filesize</th>
<th>Atime</th>
<th>Attr Flags</th>
<th>Ctime</th>
<th>Crtime</th>
<th>LSN</th>
<th>Mtime</th>
<th>Par Ref</th>
<th>Par Seq</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntio404.sys</td>
<td>mft</td>
<td>34560</td>
<td>2009-11-09T01:24:59Z</td>
<td>2080</td>
<td>2008-04-14T12:00:00Z</td>
<td>2009-11-08T17:08:04Z</td>
<td>29295332</td>
<td>2008-04-14T12:00:00Z</td>
<td>71</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Error rate for FAT32 is high; ignore these if drive is not FAT:

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Object Source</th>
<th>Filesize</th>
<th>Atime</th>
<th>Attr</th>
<th>Ctime</th>
<th>Crtime</th>
<th>LSN</th>
<th>Mtime</th>
<th>Startcluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>-eSigPol.icy</td>
<td>fat</td>
<td>1937007917</td>
<td>2037-09-13T00:00:00</td>
<td>45</td>
<td>2030-03-09T00:00:00</td>
<td>2034-09-13T12:43:13</td>
<td>1701667951</td>
<td>2010-09-29T00:00:00</td>
<td>173063680-GZIP-470016 dukdxd1o.lH7  &lt;fileobject src='fat'&gt;</td>
</tr>
<tr>
<td>dukdxd1o.lH7</td>
<td>fat</td>
<td>1632198449</td>
<td>2010-09-29T00:00:00</td>
<td>32</td>
<td>1999-09-25T06:34:01</td>
<td>2007-01-18T15:01:17</td>
<td>2016504113</td>
<td>2007-01-18T15:01:17</td>
<td>173063680-GZIP-470016 dukdxd1o.lH7  &lt;fileobject src='fat'&gt;</td>
</tr>
</tbody>
</table>
ZIP has become the *defacto* archive format.
- zip, jar, docx, pptx, etc.
- ZIP64 provides for files larger than 4GiB
- Allows faster access to components that .tar.gz

**bulk_extractor** finds local file headers.

A. Local file header:

- local file header signature: 4 bytes (0x04034b50)
- version needed to extract: 2 bytes
- general purpose bit flag: 2 bytes
- compression method: 2 bytes
- last mod file time: 2 bytes
- last mod file date: 2 bytes
- crc-32: 4 bytes
- compressed size: 4 bytes
- uncompressed size: 4 bytes
- file name length: 2 bytes
- extra field length: 2 bytes

- file name (variable size)
- extra field (variable size)
zip.txt decodes every potential header of every zip archive

# Filename: /corp/nps/scenarios/2009-m57-patents/drives-redacted/
charlie-2009-12-11.E01
# Feature-Recorder: zip
# Feature-File-Version: 1.1
62865144 000024.tif <zipinfo><name>000024.tif</name><name_len>10</name_len><version>20</version><compression_method>8</compression_method><uncompr_size>0</uncompr_size><compr_size>0</compr_size><lastmodtime>8</lastmodtime><lastmoddate>34592</lastmoddate><crc32>0</crc32><extra_field_len>0</extra_field_len><disposition bytes='10846' decompressed</disposition></zipinfo>

62874091 000025.tif <zipinfo><name>000025.tif</name><name_len>10</name_len><version>20</version><compression_method>8</compression_method><uncompr_size>0</uncompr_size><compr_size>0</compr_size><lastmodtime>8</lastmodtime><lastmoddate>34592</lastmoddate><crc32>0</crc32><extra_field_len>0</extra_field_len><disposition bytes='67680' decompressed</disposition></zipinfo>

Possible uses:
- Identify MSOffice and OpenOffice documents
- Identify Java programs
- Reconstruct hierarchy
There are four main categories of feature files:

Identity Information:
- Domain Names; Email addresses; URLs
- Search terms; Facebook IDs; JSON data
- KML files
- VCARDs
- find output

Technical Info:
- ZIP files; EXIF data

Network Information:
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.

Information about executables:
- ELF & PE headers; Windows Prefetch files
ether.txt and ether_histogram.txt: a list of potential ethernet addresses (from packets and ASCII)

<table>
<thead>
<tr>
<th>Address</th>
<th>Ethernet Address</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>342699417</td>
<td>00:80:77:31:01:07</td>
<td>n008077310107 1 00:80:77:31:01:07 192.168.1.2 an the following: 00:80:77:31:01:07 brn008077310107</td>
</tr>
<tr>
<td>567912435</td>
<td>00:80:77:31:01:07</td>
<td>-s 192.168.1.2 00:80:77:31:01:07&lt;/div&gt;</td>
</tr>
<tr>
<td>684600847</td>
<td>00:80:77:31:01:07</td>
<td>(ether_dhost)</td>
</tr>
<tr>
<td>6341279242</td>
<td>00:0B:DB:4F:6B:10</td>
<td>(ether_dhost)</td>
</tr>
<tr>
<td>6341279242</td>
<td>00:19:E3:E7:5D:23</td>
<td>(ether_shost)</td>
</tr>
<tr>
<td>6341283338</td>
<td>00:0B:DB:4F:6B:10</td>
<td>(ether_dhost)</td>
</tr>
<tr>
<td>6341283338</td>
<td>00:19:E3:E7:5D:23</td>
<td>(ether_shost)</td>
</tr>
<tr>
<td>6341287434</td>
<td>00:0B:DB:4F:6B:10</td>
<td>(ether_dhost)</td>
</tr>
</tbody>
</table>

Note:
- Packets clearly traveled from 00:19:E5:E7:5D:23 to 00:0B:DB:4F:6B:10
- Other usage appears to have Ethernet addresses in HTML!
### ip.txt: potential ip addresses from packet carving (scan_net) (not from dotted quads)

- Local (“L”) or Remote (“R”)
- chksum-bad/chksum-ok — IP checksum good or bad
- sockaddr_in — IP address from sockaddr_in structure.

<table>
<thead>
<tr>
<th>RSSI</th>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>117942521</td>
<td>20.137.78.24</td>
<td>struct ip R (src) chksum-bad</td>
</tr>
<tr>
<td>117942521</td>
<td>94.89.93.194</td>
<td>struct ip L (dst) chksum-bad</td>
</tr>
<tr>
<td>118342942</td>
<td>20.137.78.24</td>
<td>struct ip R (src) chksum-bad</td>
</tr>
<tr>
<td>118342942</td>
<td>94.89.93.194</td>
<td>struct ip L (dst) chksum-bad</td>
</tr>
<tr>
<td>9977306594</td>
<td>192.168.1.1</td>
<td>sockaddr_in</td>
</tr>
<tr>
<td>9977393926</td>
<td>63.245.209.93</td>
<td>sockaddr_in</td>
</tr>
<tr>
<td>5839793854-HIBER-17952268</td>
<td>90.4.162.232</td>
<td>struct ip L (dst) chksum-bad</td>
</tr>
<tr>
<td>5839793854-HIBER-17960460</td>
<td>78.0.3.185</td>
<td>struct ip R (src) chksum-bad</td>
</tr>
<tr>
<td>5839793854-HIBER-17960460</td>
<td>90.4.162.232</td>
<td>struct ip L (dst) chksum-bad</td>
</tr>
<tr>
<td>6339825268</td>
<td>192.168.1.104</td>
<td>struct ip L (src) chksum-ok</td>
</tr>
<tr>
<td>6339825268</td>
<td>192.168.1.1</td>
<td>struct ip R (dst) chksum-ok</td>
</tr>
<tr>
<td>6339825320</td>
<td>192.168.1.104</td>
<td>struct ip L (src) chksum-ok</td>
</tr>
<tr>
<td>5839793854-HIBER-129985200</td>
<td>8.3.2.3</td>
<td>sockaddr_in</td>
</tr>
</tbody>
</table>
ip_histogram.txt removes random noise
(1.3 histogram is only of chksum-ok values)

Histogram of all values:

```
# Filename: /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01
# Feature-Recorder: ip
# Histogram-File-Version: 1.1
n=93    108.5.218.9
n=93    7.90.102.193
n=64    20.137.78.24
n=64    94.89.93.194
n=31    176.69.248.3
n=30    5.225.0.252
n=26    120.23.102.15
n=26    182.210.102.137
n=24    152.6.0.164
n=24    152.6.0.220
n=19    192.168.1.1
n=14    192.168.1.104
n=13    141.77.252.81
n=13    80.4.139.6
```

chksum-ok:
packets.pcap — pcap file made from carved potential packets.

Use any packet analysis tool you like...

```
$ tcpdump -r packets.pcap
-5:-59:-59.0000 IP 192.168.1.1.microsoft-ds > 192.168.1.104.udpradio: Flags [.] ,
ack 416616880, win 65535, length 0
-5:-59:-59.0000 IP 192.168.1.1.microsoft-ds > 192.168.1.104.udpradio: Flags [.] ,
ack 4294967234, win 65535, length 0
-5:-59:-59.0000 IP 192.168.1.1.microsoft-ds > 192.168.1.104.udpradio: Flags [.] ,
ack 4294967084, win 65535, length 0

-5:-59:-59.0000 IP 192.168.1.1.microsoft-ds > 192.168.1.104.udpradio: Flags [P.],
seq 4294966956:4294967060, ack 4294967008, win 65535, length 104SMB PACKET:
SMBtrans2 (REPLY)
...
```

Notice time is -5:-59:-59.000

- My local time zone was -0600
- The time in the packet file is “1”
  /* Possibly a valid ethernet frame but not preceeded by a pcap_record_header.
   * Write it out with time of 1.
   */
- Only packets carved from a PCAP file will have a the correct time.
tcp.txt — Details about TCP (and UDP) network flows

More detail than ip.txt

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>117942521</td>
<td>20.137.78.24:2048 -&gt; 94.89.93.194:32824</td>
<td>TCP</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>118342942</td>
<td>20.137.78.24:2048 -&gt; 94.89.93.194:32824</td>
<td>TCP</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>119672053</td>
<td>255.144.140.1:3972 -&gt; 0.0.133.192:52224</td>
<td>TCP</td>
<td>3973</td>
<td></td>
</tr>
<tr>
<td>122908648</td>
<td>1.0.0.0:0 -&gt; 117.17.2.0:0</td>
<td>UDP</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>101356868</td>
<td>56.141.76.36:65490 -&gt; 28.81.139.206:35832</td>
<td>TCP</td>
<td>3972</td>
<td></td>
</tr>
<tr>
<td>101727492</td>
<td>56.141.76.36:65490 -&gt; 28.81.139.206:35832</td>
<td>TCP</td>
<td>3972</td>
<td></td>
</tr>
<tr>
<td>102361428</td>
<td>56.141.76.36:65490 -&gt; 28.81.139.206:35832</td>
<td>TCP</td>
<td>3972</td>
<td></td>
</tr>
<tr>
<td>102380242</td>
<td>20.137.78.24:2048 -&gt; 94.89.93.194:21899</td>
<td>TCP</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>68852207</td>
<td>7.90.102.193:13311 -&gt; 108.5.218.9:18387</td>
<td>TCP</td>
<td>3973</td>
<td></td>
</tr>
</tbody>
</table>

Be careful of false positives:

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>336089314-HIBER-100696361</td>
<td>0.0.0.0:101 -&gt; 0.0.0.0:19829</td>
<td>TCP</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-113107975</td>
<td>48.144.141.49:0 -&gt; 176.61.0.0:0</td>
<td>TCP</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-161355043</td>
<td>7.86.252.232:55425 -&gt; 47.0.250.69:21841</td>
<td>TCP</td>
<td>1419</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166154373</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166162086</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166169799</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166194316</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166202507</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166210698</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166218889</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-166227080</td>
<td>255.118.14.233:57600 -&gt; 255.164.149.80:52428</td>
<td>UDP</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-168358773</td>
<td>57.93.93.93:3968 -&gt; 8.141.88.247:17843</td>
<td>TCP</td>
<td>5631</td>
<td></td>
</tr>
<tr>
<td>336089314-HIBER-168361526</td>
<td>57.93.93.93:3968 -&gt; 8.141.88.247:17843</td>
<td>TCP</td>
<td>5631</td>
<td></td>
</tr>
</tbody>
</table>
tcp_histogram.txt — would be nice to have total flow info

These packets:

101727492  56.141.76.36:65490 -> 28.81.139.206:35832 (TCP)  Size: 3972
102361428  56.141.76.36:65490 -> 28.81.139.206:35832 (TCP)  Size: 3972

Become this histogram:

n=93    7.90.102.193:13311 -> 108.5.218.9:18387 (TCP)
n=53    0.0.123.55:12288 -> 56.49.57.65:12336 (TCP)
n=48    5.100.228.83:64 -> 15.134.211.0:15 (TCP)
n=38    252.21.212.255:34048 -> 83.0.0.0:17792 (TCP)
n=38    252.21.212.255:34048 -> 83.0.16.16:17792 (TCP)
n=30    104.48.235.16:60160 -> 232.235.16.232:35701 (TCP)
n=30    5.225.0.252:61133 -> 176.69.248.3:63488 (TCP)
n=28    0.106.37.95:23179 -> 102.59.199.117:52968 (TCP)
n=27    20.137.78.24:2048 -> 94.89.93.194:21899 (TCP)
n=26    120.23.102.15:4160 -> 182.210.102.137:16449 (UDP)
n=24    106.0.80.83:51457 -> 141.74.255.139:65382 (UDP)

Caveats:

- Still a lot of false positives.
- The current histogram system can’t do math...
There are four main categories of feature files:

**Identity Information:**
- Domain Names; Email addresses; URLs
- Search terms; Facebook IDs; JSON data
- KML files
- VCARDS
- find output

**Technical Info:**
- ZIP files; EXIF data

**Network Information:**
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.

**Information about executables:**
- ELF & PE headers; Windows Prefetch files
elf.txt records potential ELF executables

carlie-2009-12-11 doesn’t have any:

    -rw-r--r--+  1 simsong  simsong         0 Jul  20 16:54 elf.txt

But nps-2009-ubnist1.gen3 does:

    -rw-r--r--+  1 simsong  staff   5691737 Aug  3 12:39 elf.txt

Here is a sample:

```
# Feature-File-Version: 1.1
727114768-GZIP-2048   1b5984e4365278bee12c9be8849439f4        <ELF
    class="ELFCLASS32" data="ELFDATA2LSB" osabi="ELFOSABI_NONE" abiversion="0" <ehdr
type="ET_EXEC" machine="EM_386" version="1" entry="134514864" phoff="52"
shoff="19000" flags="0" ephsize="52" phentsize="32" phnum="8" shentsize="40"
shnum="27" shstrndx="26" />
<sections><section name="" type="SHT_NULL" addr="0x0"
offset="0" size="0" link="0" info="0" addralign="0" shentsize="0"/>
<flags></section><section name=".interp" type="SHT_PROGBITS" addr="0x8048134"
offset="134" size="13" link="0" info="0" addralign="1"
shentsize="0"/>
<flags><SHF_ALLOC /></flags></section><section name=".note.ABI-tag"
type="SHT_NOTE" addr="0x8048148" offset="148" size="20" link="0" info="0"
addralign="4" shentsize="0"/>
<flags><SHF_ALLOC /></flags></section><section name=".hash" type="SHT_HASH" addr="0x8048168" offset="168" size="c0" link="5"
info="0" addralign="4" shentsize="4"/>
<flags><SHF_ALLOC /></flags></section><section name=".gnu.hash" type="SHT_GNU_HASH" addr="0x8048228" offset="
```
Decoding the \texttt{<ELF>} record...

The path indicates that the ELF is inside a GZIP stream:

\begin{verbatim}
# Feature-File-Version: 1.1
727114768-GZIP-2048 ...
\end{verbatim}

The MD5 is the hash of the first 4KiB:

1b5984e4365278bee12c9be8849439f4

Next comes the XML for the header:

\begin{verbatim}
<ELF class="ELFCLASS32" data="ELFDATA2LSB" osabi="ELFOSABI_NONE" abiversion="0">
  <ehdr type="ET_EXEC" machine="EM_386" version="1" entry="134514864" phoff="52"
    shoff="19000" flags="0" ehsiz
...
WinPE.txt — Potential Windows executables
First line is the offset, MD5(first 4K), XML of data

117886464       0316eaac06e782616036639824c04ceb       <PE>
<FileHeader Machine=...>

Uses:
- Offset tells you where to find the file (most executables are not fragmented)
- MD5 can be used to deduplicate and look up in hash database
- <PE> XML block breaks out all of the PE headers.
<xml version="1.0"?>
<PE>
  <FileHeader
    Machine="IMAGE_FILE_MACHINE_I386"
    NumberOfSections="5"
    TimeDateStamp="1255540604"
    PointerToSymbolTable="0"
    NumberOfSymbols="0"
    SizeOfOptionalHeader="224"
  >
  <Characteristics>
    <IMAGE_FILE_EXECUTABLE_IMAGE/>
    <IMAGE_FILE_32BIT_MACHINE/>
    <IMAGE_FILE_DLL/>
  </Characteristics>
</FileHeader>
</PE>
<OptionalHeaderStandard
  Magic="PE32"
  MajorLinkerVersion="8"
  MinorLinkerVersion="0"
  SizeOfCode="260096"
  SizeOfInitializedData="89088"
  SizeOfUninitializedData="0"
  AddressOfEntryPoint="0x3963c"
  BaseOfCode="0x1000"/>
<OptionalHeaderWindows>
  ImageBase="0x6a520000"
  SectionAlignment="1000"
  FileAlignment="200"
  MajorOperatingSystemVersion="4"
  MinorOperatingSystemVersion="0"
  MajorImageVersion="0"
  MinorImageVersion="0"
  MajorSubsystemVersion="4"
  MinorSubsystemVersion="0"
  Win32VersionValue="0"
  SizeOfImage="59000"
  SizeOfHeaders="400"
  CheckSum="0x5aedb"
  SubSystem=""
  SizeOfStackReserve="100000"
  SizeOfStackCommit="1000"
  SizeOfHeapReserve="100000"
  SizeOfHeapCommit="1000"
  LoaderFlags="0"
  NumberOfRvaAndSizes="10">
  <DllCharacteristics/>
</OptionalHeaderWindows>
Provides details of each PE section

<Sections>
  <SectionHeader
    Name=".text"
    VirtualSize="3f73a"
    VirtualAddress="1000"
    SizeOfRawData="3f800"
    PointerToRawData="400"
    PointerToRelocations="0"
    PointerToLinenumbers="0">
    <Characteristics>
      <IMAGE_SCN_CNT_CODE/>
      <IMAGE_SCN_MEM_EXECUTE/>
      <IMAGE_SCN_MEM_READ/>
    </Characteristics>
  </SectionHeader>
  <SectionHeader
    Name=".rdata"
    VirtualSize="df22"
    VirtualAddress="41000"
    SizeOfRawData="e000"
    PointerToRawData="3fc00"
    PointerToRelocations="0"
    PointerToLinenumbers="0">
    <Characteristics>
      <IMAGE_SCN_CNT_INITIALIZED_DATA/>
      <IMAGE_SCN_MEM_READ/>
    </Characteristics>
  </SectionHeader>
</Sections>
Prefetch files give you:

- Name of executable
- Serial number
- Name of DLLs
- Directory of DLLs
- atime
- ctime
- Number of runs

62123520  WMIPRVSE.EXE  <prefetch><os>Windows XP</os>

<filename>WMIPRVSE.EXE</filename>
<header_size>152</header_size>
<atime>2009-12-11T15:31:12Z</atime>
<runs>251</runs>
<filenames>
<file>\x5CDEVICE\x5CHARDDISKVOLUME1\x5CWINDOWS\x5CSYSTEM32\x5CNTDLL.DLL</file>
<file>\x5CDEVICE\x5CHARDDISKVOLUME1\x5CWINDOWS\x5CSYSTEM32\x5CKERNEL32.DLL</file>
<file>\x5CDEVICE\x5CHARDDISKVOLUME1\x5CWINDOWS\x5CSYSTEM32\x5CUNICODE.NLS</file>
<file>\x5CDEVICE\x5CHARDDISKVOLUME1\x5CWINDOWS\x5CSYSTEM32\x5CLOCALE.NLS</file>
...
</filenames>
<volume><path>\x5CDEVICE\x5CHARDDISKVOLUME1</path>
<creation>2009-11-08T16:58:56Z</creation>
<serial_number>d8cc759a</serial_number>
<dirnames><dir>\x5CDEVICE\x5CHARDDISKVOLUME1\x5C</dir>
<dir>\x5CDEVICE\x5CHARDDISKVOLUME1\x5CWINDOWS\x5C</dir>
...
False positives must be expected.

When TB of data are scanned, there will be false positives.

These IPv6 addresses are from packets with valid checksums.
- But the IPv6 checksum is just 32 bits, so there is a 1-in-4-billion chance of a false positive.

These are probably invalid SSNs:

48857564047  SSN759057878
48858284573  SSN759057878

Objects with internal binary structures will rarely be in error.
- Windows PE files, LNK files, JPEGs, etc.
Only works with contiguous databases

For example, in testing on the drive AE10-1156, 12 sqlite databases were carved, 7 of which had recoverable schemas, and 2 of which had recoverable data:

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>33554432.sqlite3</td>
<td>9,269,760</td>
<td>moz_downloads (empty)</td>
</tr>
<tr>
<td>402653184.sqlite3</td>
<td>9,008,640</td>
<td>moz_hosts (empty)</td>
</tr>
<tr>
<td>452984832.sqlite3</td>
<td>6,022,656</td>
<td>engine_data (empty)</td>
</tr>
<tr>
<td>469762048.sqlite3</td>
<td>8,785,408</td>
<td>moz_cookies (7 cookies)</td>
</tr>
<tr>
<td>486539264.sqlite3</td>
<td>10,107,392</td>
<td>moz_logins (empty)</td>
</tr>
<tr>
<td>570425344.sqlite3</td>
<td>4,379,136</td>
<td>moz groups, prefs, settings (empty)</td>
</tr>
<tr>
<td>587202560.sqlite3</td>
<td>8,445,440</td>
<td>moz bookmarks, history, keywords, etc. (98 bookmarks</td>
</tr>
<tr>
<td>822083584.sqlite3</td>
<td>3,988,992</td>
<td>moz classifier, subs and tables _classifier,</td>
</tr>
<tr>
<td>1426063360.sqlite3</td>
<td>8,258,804</td>
<td>(nothing recoverable)</td>
</tr>
<tr>
<td>2684354560.sqlite3</td>
<td>12,385,876</td>
<td>(nothing recoverable)</td>
</tr>
<tr>
<td>2717908992.sqlite3</td>
<td>178,688</td>
<td>(nothing recoverable)</td>
</tr>
<tr>
<td>2936012800.sqlite3</td>
<td>16,983,284</td>
<td>(nothing recoverable)</td>
</tr>
</tbody>
</table>
Finding File Names
identify_filenames.py: Determines the file name for each feature.

bulk_extractor reports the *offset in the disk image* for each feature.

To get the file names, you need to map the disk block to a file.

- Make a map of the blocks in DFXML with fiwalk ([https://github.com/kfairbanks/sleuthkit](https://github.com/kfairbanks/sleuthkit)) — Soon to be integrated into SleuthKit
- Then use python/identify_filenames.py to create an annotated feature file.

*identify_filenames* correlation the feature file and the DFXML file!
Identify filenames from "bulk_extractor" output

Positional arguments:
- `bulk_extractor_output`: Directory or ZIP file of bulk_extractor output
- `outdir`: Output directory; must not exist

Optional arguments:
- `-h, --help`: Show this help message and exit
- `--all`: Process all feature files
- `--featurefiles FEATUREFILES`: Specific feature file to process; separate with commas
- `--imagefile IMAGEFILE`: Overwrite location of image file from bulk_extractor output
- `--xmlfile XMLFILE`: Don't run fiwalk; use the provided XML file instead
- `--list`: List feature files in bulk_extractor_output and exit
- `-t`: Terse output
- `-v`: Print Version and exit
- `--verbose`: Verbose mode
- `--debug`: Debug mode
identify_filenames.py tries to use the information in the report.xml file to make operation automatic.

report.xml is a DFXML file that contains:
- Disk image that was processed
- Location of feature files

identify_filenames can work with:
- bulk_extractor output file
- a ZIP of a bulk_extractor output file
- disk image or DFXML of disk image

identify_filenames will run fiwalk if...
- no XML file is provided
- fiwalk is in the path
- But it's faster to provide the XML file!

The DFXML file has to be re-read for each feature file.
identify_filenames typically takes hours to run.

Time is proportional to (# of features) * (# of file fragments)

    $ python3.2 python/identify_filenames.py ~/charlie-2009-12-10.zip
    charlie-2009-12-10-id2 --xmlfile charlie-2009-12-10.xml --all
    Adding features from aes_keys.txt
    Using XML file /corp/nps/scenarios/2009-m57-patents/drives_dfxml/charlie-2009-12-10.xml
    Processed 1000 fileobjects in DFXML file
    ...
    Processed 39000 fileobjects in DFXML file
    Processed 40000 fileobjects in DFXML file
    Generating output...
    real  10298.68
    user  10286.50
    sys   8.25
    $

Roughly 3 hours for a 60GB disk image.
Output is “annotated” feature files.

$ ls -l
total 166088
-rw-r--r--+ 1 simsong  simsong  511 Aug  4 18:04 annotated_aes_keys.txt
-rw-r--r--+ 1 simsong  simsong  3511 Aug  4 15:39 annotated_ccn.txt
-rw-r--r--+ 1 simsong  simsong 24986176 Aug  4 17:53 annotated_domain.txt
-rw-r--r--+ 1 simsong  simsong 1882453 Aug  4 18:03 annotated_email.txt
-rw-r--r--+ 1 simsong  simsong   24451 Aug  4 16:48 annotated_ether.txt
-rw-r--r--+ 1 simsong  simsong 11208045 Aug  4 15:39 annotated_exif.txt
-rw-r--r--+ 1 simsong  simsong  125580 Aug  4 18:03 annotated_ip.txt
-rw-r--r--+ 1 simsong  simsong  3465286 Aug  4 21:40 annotated_json.txt
-rw-r--r--+ 1 simsong  simsong  3823218 Aug  4 18:26 annotated_rfc822.txt
-rw-r--r--+ 1 simsong  simsong   268678 Aug  4 21:41 annotated_tcp.txt
-rw-r--r--+ 1 simsong  simsong    79345 Aug  4 21:42 annotated_telephone.txt
-rw-r--r--+ 1 simsong  simsong  69150534 Aug  4 16:48 annotated_url.txt
-rw-r--r--+ 1 simsong  simsong  18776356 Aug  4 18:00 annotated_windirs.txt
-rw-r--r--+ 1 simsong  simsong  1944968 Aug  4 22:15 annotated_winprefetch.txt
-rw-r--r--+ 1 simsong  simsong  34263928 Aug  4 18:20 annotated_zip.txt
$
<table>
<thead>
<tr>
<th># Position</th>
<th>Feature</th>
<th>Context</th>
<th>Filename</th>
<th>File MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>7277995794</td>
<td>4857994530998756</td>
<td>ible-price/&amp;rnd=4857994530998756\x00request-method\x00 Documents and Settings/Charlie/Local Settings/Application Data/Mozilla/Firefox/Profiles/2usvf7i1.default/Cache/<em>CACHE_001</em></td>
<td>eca068c08645e300edd7530362d80a97</td>
<td></td>
</tr>
</tbody>
</table>

- position: 7277995794
- Feature: 4857994530998756
- Context: ible-price/\&rnd=4857994530998756\x00request-method\x00 Documents and Settings/Charlie/Local Settings/Application Data/Mozilla/Firefox/Profiles/2usvf7i1.default/Cache/_CACHE_001_
- Filename: Documents and Settings/Charlie/Local Settings/Application Data/Mozilla/Firefox/Profiles/2usvf7i1.default/Cache/_CACHE_001_
- File MD5: eca068c08645e300edd7530362d80a97

<table>
<thead>
<tr>
<th>3598712863-ZIP-100622</th>
<th><a href="mailto:michael.buettner@sun.com">michael.buettner@sun.com</a></th>
<th>chael Büttner <a href="mailto:michael.buettner@sun.com">michael.buettner@sun.com</a>\x0A</th>
<th>Philipp Documents and Settings/Charlie/My Documents/Downloads/lightning-0.9-tb-win.xpi</th>
<th>70eefacfe1227e50db99556cf98161e</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>position:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3598712863-ZIP-100622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:michael.buettner@sun.com">michael.buettner@sun.com</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Context:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>chael Büttner <a href="mailto:michael.buettner@sun.com">michael.buettner@sun.com</a>\x0A</td>
<td>Philipp</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Filename:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Documents and Settings/Charlie/My Documents/Downloads/lightning-0.9-tb-win.xpi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>File MD5:</td>
<td></td>
<td></td>
<td>70eefacfe1227e50db99556cf98161e</td>
</tr>
</tbody>
</table>
Getting more information

Bulk_Extractor:
- Programmer’s Manual
- User’s Manual

See also: