Using bulk_extractor for digital forensics triage and cross-drive analysis

Simson L. Garfinkel

- Wednesday, August 8th, 2012. 12:30–2:30

- This tutorial will provide an in-depth introduction to the use of bulk_extractor, a high-speed feature extractor tool that can be used with any kind of digital forensics data. The tutorial will discuss how to use bulk_extractor for rapid triage of new media, how to use bulk_extractor's post-processing features for file identification and cross-drive correlation, and how to tune bulk_extractor to improve performance. Finally the internal design of the program will be presented, with instructions on how to develop new bulk_extractor modules.
NPS is the Navy’s Research University.

Location: Monterey, CA

Students: 1500
- US Military (All 5 services)
- US Civilian (Scholarship for Service & SMART)
- Foreign Military (30 countries)

Schools:
- Business & Public Policy
- Engineering & Applied Sciences
- Operational & Information Sciences
- International Graduate Studies

NCR Initiative:
- 8 offices on 5th floor, 900N Glebe Road, Arlington
- Current staffing: 4 professors, 2 lab managers, 2 programmers, 4 contractors
- OPEN SLOTS FOR .GOV PHDs!
NPS research: “Automated Media Exploitation”

Area #1: Bulk Data Analysis
- Feature extraction (bulk_extractor)
- Statistical techniques (random sampler)
- Similarity metrics (sdhash & sdtext)

Area #2: End-to-end automation of forensic processing
- Digital Forensics XML Toolkit (fiwalk, md5deep, etc.)
- Disk Image \(\Rightarrow\) Power Point (smirk)

Area #3: Data mining for digital forensics
- Automated analysis (cross-drive analysis)

Area #4: Creating Standardized Forensic Corpora
- Freely redistributable disk and memory images, packet dumps, files (digitalcorpora.org).
Outline of today’s tutorial

Introducing bulk_extractor
  - Overview and history
  - Output formats

Using bulk_extractor’s output

Finding files

Cross drive analysis

Internal structure and writing plug-ins
Introducing bulk_extractor
Stream-Based Disk Forensics:
Scan the disk from beginning to end; do your best.

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.

0 1TB
3 hours, 20 min to read the data
Primary Advantage: Speed

No disk seeking! (Good for HDs, SSDs, & E01 files)
Easy to parallelize (“embarrassingly parallel”)
Reads all the data — allocated files, deleted files, file fragments

Caveats:
- Compressed data must be decompressed
  — *Fragmented, compressed files may not be recovered*
- Can only read at maximum I/O transfer rates if data can be *processed*
  — *Even 24+ cores may not be enough*
- Does not provide file names
  — *File names can be determined with a separate metadata extraction step.*
Fragmented files may not be recovered

ZIP, GZIP & LZMA use *adaptive* compression algorithms.
- Part 1 required to decompress part 2.
- Also an issue for JPEG.

Fortunately, most files are *not* fragmented.
- Individual components of a ZIP can be recovered (e.g. `word/document.xml`)

Most files that *are* fragmented have carvable internal structure:
- Log files, Outlook PST files, etc.
Our experience:
bulk_extractor is faster and finds data other tools miss.

Runs 2-10 times faster than EnCase or FTK on the same hardware.
- bulk_extractor is multi-threaded; EnCase 6.x and FTK 3.x have little threading.

Finds email address, URLs, CCNs that other tools miss
- “Optimistically” decompresses and re-analyzes all data.
  —zip fragments
  —gzip browser cache runs
- Decompression operates on incomplete and corrupted data (until decompression fails)
- Decompresses fragments of Windows Hibernation Files
- Builds word lists for password cracking
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 process with bulk_extractor_reader.py
Feature files are UTF-8 files that contain extracted data.

```
# 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/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
...
3772517888-GZIP-28322  (831) 373-5555  onterey-<nobr>(831) 373-5555</nobr>
3772517888-GZIP-29518  (831) 899-8300  Seaside - <nobr>(831) 899-8300</nobr>
5054604751    716-871-2929    a%,888-571-2048,716-871-2929\x0D\x0ACPV,,,,%Cape
```

Designed for easy processing by python, perl or C++ program

- “Loosely ordered.”
- -GZIP- indicates that data was decompressed
- Non-UTF-8 characters are escaped
Histogram system automatically summarizes features.

# 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/charlie-2009-12-11.E01
# Feature-Recorder: email
# Histogram-File-Version: 1.1
...
n=875 mozilla@kewis.ch (utf16=3)
n=651 charlie@m57.biz (utf16=120)
n=605 ajbanck@planet.nl
...
n=288 mattwillis@gmail.com
n=281 garths@oeone.com
n=226 michael.buettner@sun.com (utf16=2)
n=225 bugzilla@babylonsounds.com
n=218 berend.cornelius@sun.com
n=210 ips@mail.ips.es
n=201 mschroeder@mozilla.x-home.org
n=186 pat@m57.biz (utf16=1)
Histogram of search terms can convey intent.

<table>
<thead>
<tr>
<th>Term</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>exotic+car+dealer</td>
<td>53</td>
</tr>
<tr>
<td>ford+car+dealer</td>
<td>41</td>
</tr>
<tr>
<td>2009+Shelby</td>
<td>34</td>
</tr>
<tr>
<td>steganography</td>
<td>25</td>
</tr>
<tr>
<td>General+Electric</td>
<td>23</td>
</tr>
<tr>
<td>time+travel</td>
<td>23</td>
</tr>
<tr>
<td>steganography+tool+free</td>
<td>19</td>
</tr>
<tr>
<td>vacation+packages</td>
<td>19</td>
</tr>
<tr>
<td>firefox</td>
<td>16</td>
</tr>
<tr>
<td>quicktime</td>
<td>16</td>
</tr>
<tr>
<td>7zip</td>
<td>14</td>
</tr>
<tr>
<td>fox+news</td>
<td>14</td>
</tr>
<tr>
<td>hex+editor</td>
<td>13</td>
</tr>
</tbody>
</table>
New in bulk_extractor 1.3

New supported data types:
- Windows PE Scanner
- Linux ELF Scanner
- VCF Card Scanner
- BASE16 scanner
- Windows directory carver

New Histogram options:
- Numeric only option for phone numbers
- Supports new Facebook ID

Better Unicode Support:
- Histograms now UTF-8 / UTF-16 aware
- Feature files are UTF-8 clean

Limited support for file carving:
- Packets carved into pcap files
- VCF Card carver
A bulk_extractor success story:  
City of San Luis Obispo Police Department, Spring 2010

District Attorney filed charges against two individuals:
- Credit Card Fraud
- Possession of materials to commit credit card fraud.

Defendants:
- Arrested with a computer.
- Expected to argue that defendants were unsophisticated and lacked knowledge.

Examiner given 250GB drive *the day before preliminary hearing*.
- Typically, it would take several days to conduct a proper forensic investigation.
bulk_extractor found actionable evidence in 2.5 hours!

Bulk_extractor found:
- Over 10,000 credit card numbers on the HD (1000 unique)
- Most common email address belonged to the primary defendant (possession)
- The most commonly occurring Internet search engine queries concerned credit card fraud and bank identification numbers (intent)
- Most commonly visited websites were in a foreign country whose primary language is spoken fluently by the primary defendant.

Armed with this data, the defendants were held without bail.
In 1991 I developed SBook, a free-format address book.

SBook automatically found addresses, phone numbers, email addresses *while you typed.*
Today we call this technology Named Entity Recognition

SBook’s technology was based on:

- Regular expressions executed in parallel
  — *US, European, & Asian Phone Numbers*
  — *Email Addresses*
  — *URLs*
- A gazette with more than 10,000 names:
  — *Common “Company” names*
  — *Common “Person” names*
  — *Every country, state, and major US city*
- Hand-tuned weights and additional rules.

Implementation:

- 2500 lines of GNU flex, C++
- 50 msec to evaluate 20 lines of ASCII text.
  — *Running on a 25Mhz 68030 with 32MB of RAM!*
In 2003, I bought 200 used hard drives

The goal was to find drives that had not been properly sanitized.

First strategy:
- `dd` all of the disks to image files
- `strings` to extract printable strings.
- `grep` to scan for email, CCN, etc.
  - VERY SLOW!!!!
  - HARD TO MODIFY!

Second strategy:
- Use SBook technology!
- Read disk 1MB at a time
- Pass the *raw disk sectors* to flex-based scanner.
- Big surprise: scanner didn’t crash!
flex-based scanners required substantial post-processing to be useful

- Additional validation beyond regular expressions (CCN Luhn algorithm, etc).
- Examination of feature “neighborhood” to eliminate common false positives.
- Counting and histogram analysis to find drives with most “credit card numbers”

The technique worked well to find drives with sensitive information.
Between 2005 and 2008, we interviewed law enforcement regarding their use of forensic tools.

Law enforcement officers wanted a *highly automated* tool for finding:
- Email addresses & Credit card numbers (including track 2 information)
- Phone numbers, GPS coordinates & EXIF information from JPEGs
- Search terms (extracted from URLs)
- All words that were present on the disk (for password cracking)

The tool had to:
- Run on Windows, Linux, and Mac-based systems
- Run with *no* user interaction
- Operate on raw disk images, split-raw volumes, E01 files, and AFF files
- Run at maximum I/O speed of physical drive
- Never crash

Moving technology from the lab to the field has been challenging:
- Must work with evidence files of *any size* and on *limited hardware*.
- Users can't provide their data when the program crashes.
- Users are *analysts* and *examiners*, not engineers.
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

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
- AFF
- raw
- split raw
- individual disk files

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 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.
  — *Typical margin: 1MB*

Entire system is automatic:
- Image_process iterator makes *sbuf_t* buffers.
- Each buffer is processed by every scanner
- Features are automatically combined.
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>Feature</th>
<th>Feature in Evidence Context</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>tp1.username&quot;, &quot;<a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a>&quot;);__user_pref(&quot;</td>
</tr>
</tbody>
</table>
Feature histograms are created at the end of processing. They are a powerful tool for understanding evidence.

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>Frequency</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 features or regular expression extracts from features.

Simple histogram based on feature:

<table>
<thead>
<tr>
<th>n</th>
<th>Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>579</td>
<td><a href="mailto:domexuser1@gmail.com">domexuser1@gmail.com</a></td>
</tr>
<tr>
<td>432</td>
<td><a href="mailto:domexuser2@gmail.com">domexuser2@gmail.com</a></td>
</tr>
<tr>
<td>340</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>244</td>
<td><a href="mailto:CPS-requests@verisign.com">CPS-requests@verisign.com</a></td>
</tr>
<tr>
<td>242</td>
<td><a href="mailto:someone@example.com">someone@example.com</a></td>
</tr>
</tbody>
</table>

Based on regular expression extraction:

- For example, extract search terms with 
  \*search\.*q=(.*)

<table>
<thead>
<tr>
<th>n</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>pidgin</td>
</tr>
<tr>
<td>10</td>
<td>hotmail+thunderbird</td>
</tr>
<tr>
<td>3</td>
<td>Grey+Gardens+cousins</td>
</tr>
<tr>
<td>3</td>
<td>dvd</td>
</tr>
<tr>
<td>2</td>
<td>%TERMS%</td>
</tr>
<tr>
<td>2</td>
<td>cache:</td>
</tr>
<tr>
<td>2</td>
<td>p</td>
</tr>
<tr>
<td>2</td>
<td>pi</td>
</tr>
<tr>
<td>2</td>
<td>pid</td>
</tr>
<tr>
<td>1</td>
<td>Abolish+income+tax</td>
</tr>
<tr>
<td>1</td>
<td>Brad+and+Angelina+nanny+help</td>
</tr>
<tr>
<td>1</td>
<td>Build+Windmill</td>
</tr>
<tr>
<td>1</td>
<td>Carol+Alt</td>
</tr>
</tbody>
</table>
bulk_extractor has *multiple* feature extractors. Each scanner runs in order. (Order doesn’t matter.)

Scanners can be turned on or off
- Useful for debugging.
- AES key scanner is *very slow* (off by default)

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!*

Recursion used for:
- Decompressing ZLIB, Windows HIBERFILE,
- Extracting text from PDFs
- Handling compressed browser cache data
Recursion requires a *new way* to describe offsets. *bulk_extractor* introduces the “forensic path.”

Consider an HTTP stream that contains a GZIP-compressed email:

We can represent this as:

```
11052168704-GZIP-3437 live.com eMn='domexuser1@live.com';var srf_sDispM
11052168704-GZIP-3475 live.com pMn='domexuser1@live.com';var srf_sPreCk
11052168704-GZIP-3512 live.com eCk='domexuser1@live.com';var srf_sFT='<
```
Integrated design, but compact.
20,573 lines of code; 41.94 seconds to compile on i5 iMac
BEViewer: GUI runs on Windows, Mac & Linux
Launches bulk_extractor; views results

Uses bulk_extractor to decode forensic path
Running bulk_extractor
$ 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 COMPILED IN
SUPPORT FOR AFF FILES COMPILED IN
EXIV2 COMPILED IN
-o outdir - specifies output directory. Must not exist.
bulk_extractor creates this directory.

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 occurances of <regex>; may be repeated.
results go into find.txt
-q nn - Quiet Rate; only print every nn status reports. Default 0; -1 for no status

Tuning parameters:
-C NN - specifies the size of the context window (default 16)
-G NN - specify the page size (default 4194304)
-g NN - specify margin (default 4194304)
-W n1:n2 - Specifies minimum and maximum word size
(default is -w6:14)
-B NN - Specify the blocksize for bulk data analysis (default 512)
-j NN - Number of threads to run (default 8)
-M nn - sets max recursion depth (default 5)

Path Processing Mode:
-p <path>/f - print the value of <path> with a given format.
formats: r = raw; h = hex.
Specify -p - for interactive mode.
Specify -p -http for HTTP mode.

Parallelizing:
-Y <o1> - Start processing at o1 (o1 may be 1, 1K, 1M or 1G)
-Y <o1>-<o2> - Process o1-o2
-A <off> - Add <off> to all reported feature offsets

Debugging:
-h - print this message
-H - print detailed info on the scanners
-V - print version number
-z nn - start on page nn
dN - debug mode (see source code
-Z - zap (erase) output directory

Control of Scanners:
P <dir> - Specifies a plugin directory
-E scanner - turn off all scanners except scanner
-m <max> - maximum number of minutes to wait for memory starvation
    default is 60
-s name=value - sets a bulk extractor option name to be value
-e bulk - enable scanner bulk
-e exiv2 - enable scanner exiv2
-e wordlist - enable scanner wordlist
-x accts - disable scanner accts
-x aes - disable scanner aes
-x base16 - disable scanner base16
-x base64 - disable scanner base64
-x elf - disable scanner elf
-x email - disable scanner email
-x exif - disable scanner exif
-x gps - disable scanner gps
-x gzip - disable scanner gzip
-x hiber - disable scanner hiber
-x json - disable scanner json
-x kml - disable scanner kml
-x net - disable scanner net
-x pdf - disable scanner pdf
-x vcard - disable scanner vcard
-x windirs - disable scanner windirs
-x winpe - disable scanner winpe
-x winprefetch - disable scanner winprefetch
-x zip - disable scanner zip

$
bulk_extractor input and output functions

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 compiled
- 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))s (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
- **-q nn** - Quiet Rate; only print every nn status reports. Default 0;
  -1 for no status

- **-b** — Controls output appearance
- **-r, -w** — Controls which features go to which feature files
- **-F, -f** — Allows command-line find
- **-q** — Controls console output
bulk_extractor tuning

These parameters control the window and context:

Tuning parameters:
- **-C NN** - specifies the size of the context window (default 16)
- **-G NN** - specify the page size (default 4194304)
- **-g NN** - specify margin (default 4194304)
- **-W n1:n2** - Specifies minimum and maximum word size (default is -w6:14)
- **-B NN** - Specify the blocksize for bulk data analysis (default 512)
- **-M nn** - sets max recursion depth (default 5)
Normally bulk_extractor will run one analysis thread per core:

-\( j\) \( NN\)  - 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
bulk_extractor path processing is used by the GUI for printing compressed regions...

Path Processing Mode:
- \( p \) <path>/f  - print the value of <path> with a given format.
  formats: r = raw; h = hex.
Specify \(-p -\) for interactive mode.
Specify \(-p -http\) for HTTP mode.

Consider:

```
340731773-GZIP-9287   tzeruch@ceddec.com   Tom Zerucha\09\09   tzeruch@ceddec.com
\x0ATomas Fasth\09\09
```

```
$ bulk_extractor -p 340731773 /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen3.E01
1f 8b 08 08 3f 23 90 48 02 03 54 48 41 4e 4b 53 00 6d 5a 4d 73 db 38 d2 3e 4f 02 b7 bd c4 d8
$ bulk_extractor -p 340731773-GZIP-9200 /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen3.E01
```

```
$ bulk_extractor -p 340731773-GZIP-9200 /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen3.E01

bulk_extractor -p allows you to examine as much as you wish...

Satoshi.Togawa@jp.yokogawa.com. Tom Spindler... dogcow@home.merit.edu.Tom Zerucha.... tzeruch@c eddec.com.Tomas Fasth... tomas. fasth@winspot.net.Tommil Komulainen... Tommi.Komulainen@i ki.fi.Thomas Klausner .. wiza@nbnala.ifoer.tuwien.ac.at.Tomasz Kozlowski .. tomek@rentec .com.Thomas Mikkelson. tbm@ima ge.dk.Ulf Miller.... 3umo@e linormatik.uni-hamburg.de.Urko Lus a.. ulusa@euskalnet.net.Vincen t P Broman broman@spaw ar.navy.mil.Volker Quetschke quetschke@scytek.de.W Lew is wim1@hhhh..org.Walter Hofmann.... Walter.Ho ffmann@physik.stud.uni-erlangen.de.Walter Koch.. koch@hsp.de.Wa yne Chapeskie ... waynesc@spinnak er.com.Werner Koch.. wk@gnug .org.Wim Vanpeutte.. bun@bunf.p tile.rug.ac.be.Winona Brown win@huh.org.Yosiaki IIDA... iida@ring.gr.jp.Yoshihir o Kajiki. kajiki@yulog.org.... nbeck@hns.com.Thanks to the German Unix User Group for sponsoring this project, Martin Hami lton for hosting the first mailing list and OpenIT for cheap host ing conditions..The development of this software has partly be en funded by the German.Ministry for Economics and Technology under grant VIB3-68553.168-001/1999...Many thanks to my wife Gerli nde for having so much patience with me while hacking late in th e evening... Copyright 1998, 1999, 2000, 2001, 2002, 2003,
bulk_extractor “Parallelizing” options are experimental options for use with Hadoop...

Parallelizing:
- `Y <o1>`  - Start processing at o1 (o1 may be 1, 1K, 1M or 1G)
- `Y <o1>-<o2>`  - Process o1-o2
- `A <off>`  - Add <off> to all reported feature offsets

You can run multiple copies of bulk_extractor on different machines...
- But currently there is no way to easily recombine the results.
bulk_extractor debugging features

You probably won’t use these functions.... but I do.

Debugging:
- `h` - print this message
- `H` - print detailed info on the scanners
- `V` - print version number
- `z nn` - start on page nn
- `dN` - debug mode (see source code
- `Z` - zap (erase) output directory

New with version 1.3, each scanner has metadata which `-H` reports:

```
$ bulk_extractor -H

Scanner Name: accts
flags:   NONE
Scanner Interface version: 1
Author: Simson L. Garfinkel
Description: scans for CCNs, track 2, and phone #s
URL:
Scanner Version: 1.0
Feature Names: alerts ccn ccn_track2 telephone
```
bulk_extractor scanners can be compiled in or loaded on demand.

Control of Scanners:
- **P <dir>**  - Specifies a plugin directory
- **E scanner**  - turn off all scanners except scanner
- **m <max>**  - maximum number of minutes to wait for memory starvation
  default is 60

The -s option allows scanners to have settable tuning parameters:
- **-s name=value**  - sets a bulk extractor option name to be value
- **-e bulk**  - enable scanner bulk
Finally, individual scanners can be enabled or disabled.

-e enables scanners that are disabled by default:
  -e bulk - enable scanner bulk
  -e exiv2 - enable scanner exiv2
  -e wordlist - enable scanner wordlist

-x disables scanners that are enabled by default:
  -x accts - disable scanner accts
  -x aes - disable scanner aes
  -x base16 - disable scanner base16
  -x base64 - disable scanner base64
  -x elf - disable scanner elf
  -x email - disable scanner email
  -x exif - disable scanner exif
  -x gps - disable scanner gps
  -x gzip - disable scanner gzip
  -x hiber - disable scanner hiber
  -x json - disable scanner json
  -x kml - disable scanner kml
  -x net - disable scanner net
  -x pdf - disable scanner pdf
  -x vcard - disable scanner vcard
  -x windirs - disable scanner windirs
  -x winpe - disable scanner winpe
  -x winprefetch - disable scanner winprefetch
  -x zip - disable scanner zip

Don’t assume that you should enable every scanner!
<table>
<thead>
<tr>
<th>Permissions</th>
<th>Owner</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>476</td>
<td>Jul 7</td>
<td>23:50 aes_keys.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>2743</td>
<td>Jul 7</td>
<td>23:59 ccn.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>454</td>
<td>Jul 8</td>
<td>00:03 ccn_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 ccn_track2.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8</td>
<td>00:03 ccn_track2_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>23369167</td>
<td>Jul 8</td>
<td>00:03 domain.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>185266</td>
<td>Jul 8</td>
<td>00:03 domain_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 elf.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>1719842</td>
<td>Jul 8</td>
<td>00:03 email.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>35073</td>
<td>Jul 8</td>
<td>00:03 email_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>23961</td>
<td>Jul 8</td>
<td>00:00 ether.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>337</td>
<td>Jul 8</td>
<td>00:03 ether_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>11188830</td>
<td>Jul 8</td>
<td>00:03 exif.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 find.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>1112</td>
<td>Jul 8</td>
<td>00:01 gps.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 hex.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>95835</td>
<td>Jul 8</td>
<td>00:03 ip.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>11603</td>
<td>Jul 8</td>
<td>00:03 ip_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>2025702</td>
<td>Jul 8</td>
<td>00:03 json.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 kml.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>194991</td>
<td>Jul 8</td>
<td>00:03 packets.pcap</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>21343</td>
<td>Jul 8</td>
<td>00:03 report.xml</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>3782598</td>
<td>Jul 8</td>
<td>00:03 rfc822.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>213746</td>
<td>Jul 8</td>
<td>00:03 tcp.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>61255</td>
<td>Jul 8</td>
<td>00:03 tcp_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>59469</td>
<td>Jul 8</td>
<td>00:03 telephone.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>6612</td>
<td>Jul 8</td>
<td>00:03 telephone_histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>67205326</td>
<td>Jul 8</td>
<td>00:03 url.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8</td>
<td>00:03 url_facebook-id.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>5706665</td>
<td>Jul 8</td>
<td>00:03 url Histogram.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 8</td>
<td>00:03 url_microsoft-live.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>8504</td>
<td>Jul 8</td>
<td>00:03 url_searches.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>151673</td>
<td>Jul 8</td>
<td>00:03 url_services.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>0</td>
<td>Jul 7</td>
<td>23:48 vcard.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>18549729</td>
<td>Jul 8</td>
<td>00:03 windirs.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>29051041</td>
<td>Jul 8</td>
<td>00:03 winpe.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>1984759</td>
<td>Jul 8</td>
<td>00:03 winprefetch.txt</td>
</tr>
<tr>
<td>-rw-r--r--</td>
<td>simsong</td>
<td>staff</td>
<td>34128889</td>
<td>Jul 8</td>
<td>00:03 zip.txt</td>
</tr>
</tbody>
</table>
These data are from the M57 Patents Scenario

2009-M57-Patents

The 2009-M57-Patents scenario tracks the first four weeks of corporate history of the M57 Patents company. The company started operation on Friday, November 13th, 2009, and ceased operation on Saturday, December 12, 2009. As might be imagined in the business of outsourced patent searching, lots of other activities were going on at M57-Patents.

Two ways of working the scenario are as a disk forensics exercise (students are provided with disk images of all the systems as they were on the last day) and as a network forensics exercise (students are provided with all of the packets in and out of the corporate network). The scenario data can also be used to support computer forensics research, as the hard drive of each computer and each computer’s memory were imaged every day.

Instructor Materials and Answer Keys (encrypted):
- m57-instructor-packet.pdf
- hash-sets.zip
- scenario-emails.zip

Exercise slides:
digitalcorpora.org/archives/181
bulk_extractor was run on all the images in the corpora. (Was run on the redacted drive images.)

```bash
$ ls -l charlie-2009-12-11.E01
-rw-r--r--+ 1 simsong  staff  3874203396 Aug  8  2011 charlie-2009-12-11.E01

$ ewfinfo 2009-m57-patents/drives-redacted/charlie-2009-12-11.E01
ewfinfo 20120304

Acquiry information

Acquisition date: Tue Jan 11 23:49:15 2011
System date: Tue Jan 11 23:49:15 2011
Operating system used: Linux
Software version used: 20100226
Password: N/A

Media information

Media type: fixed disk
Is physical: yes
Bytes per sector: 512
Number of sectors: 19999728
Media size: 9.5 GiB (10239860736 bytes)

Digest hash information

MD5: 0377b3d41bbbc295a1c9f00aa07ee174

$
charlie-2009-12-11.zip contains the output of running bulk_extractor on the charlie-2009-12-11 disk image.

```
$ bulk_extractor -o charlie-2009-12-11 drives-redacted/charlie-2009-12-11.E01
...
$ ls -l
total 195000
-rw-r--r--+ 1 simsong  simsong       499 Jul 20 16:55 aes_keys.txt
-rw-r--r--+ 1 simsong  simsong         0 Jul 20 16:54 alerts.txt
-rw-r--r--+ 1 simsong  simsong      2668 Jul 20 17:01 ccn.txt
-rw-r--r--+ 1 simsong  simsong       477 Jul 20 17:03 ccn_histogram.txt
-rw-r--r--+ 1 simsong  simsong         0 Jul 20 17:03 ccn_track2.txt
-rw-r--r--+ 1 simsong  simsong         0 Jul 20 17:03 ccn_track2_histogram.txt
-rw-r--r--+ 1 simsong  simsong  23368758 Jul 20 17:03 domain.txt
-rw-r--r--+ 1 simsong  simsong     185281 Jul 20 17:03 domain_histogram.txt
-rw-r--r--+ 1 simsong  simsong         0 Jul 20 16:54 elf.txt
-rw-r--r--+ 1 simsong  simsong    1719865 Jul 20 17:03 email.txt
-rw-r--r--+ 1 simsong  simsong     34866 Jul 20 17:03 email_histogram.txt
...
$ egrep 'threads|clocktime' report.xml
  <threads>16</threads>
  <clocktime>537.294874</clocktime>
```

Notice:
- 195MB of output from a 40GB disk image.
- Some output files are very large, some are small.
- 16 threads required 9 minutes to run
- empty file means nothing found.
Making sense of all this data is hard!

It’s ordered alphabetically; some of the output is “experimental.”
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
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 JavaeScript
  — 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...

```
# 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: ccn
# Histogram-File-Version: 1.1
n=20   6543210123456788
n=2    4015751530102097
n=2    4920919202474441
n=1    4857994530998756
n=1    4909616081396134
n=1    5235714985079914
n=1    5273347458642687
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 “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 “domains” and host names that were found. Sources include URLs, email, dotted quads.

```
50395405    \x00h\x00o\x00t\x00m\x00a\x00i\x001\x00.\x00c\x00o\x00m\x00
\x00b\x00e\x00\x001\x002\x003\x00\x00h\x00o\x00m\x00a\x00i\x001\x00.
\x00c\x00o\x00m\x00A\x009\x00m\x00i\x00n\x00o\x00m\x00b\x00


257091    www.DocURL.com    _404.htm#http://www.DocURL.com/bar.htm \x0D\x0A\x0D\x0A

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, Bakersfield

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

27766700    205.155.65.61    ustang.nps.edu [205.155.65.61]\x0D\x0A\x09(using
27766902    m57.biz \x0D\x0A\x09for <charlie@m57.biz>; Mon, 16 Nov 2
```

Note:
- UTF-16 is “escaped” as Python-style — \x00h\x00o\x00t\x00m means “hot”
- Domains are common in compressed data
domain_histogram.txt is a hitogram of the domains...

<table>
<thead>
<tr>
<th>Domain</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org">www.w3.org</a></td>
<td>10749</td>
</tr>
<tr>
<td>chroniclingamerica.loc.gov</td>
<td>6670</td>
</tr>
<tr>
<td>openoffice.org</td>
<td>6384</td>
</tr>
<tr>
<td><a href="http://www.uspto.gov">www.uspto.gov</a></td>
<td>5998</td>
</tr>
<tr>
<td><a href="http://www.mozilla.org">www.mozilla.org</a></td>
<td>5733</td>
</tr>
<tr>
<td><a href="http://www.osti.gov">www.osti.gov</a></td>
<td>5212</td>
</tr>
<tr>
<td><a href="http://www.microsoft.com">www.microsoft.com</a></td>
<td>4952</td>
</tr>
<tr>
<td>patft.uspto.gov</td>
<td>4474</td>
</tr>
<tr>
<td><a href="http://www.gpo.gov">www.gpo.gov</a></td>
<td>4468</td>
</tr>
<tr>
<td><a href="http://www.verisign.com">www.verisign.com</a></td>
<td>3653</td>
</tr>
<tr>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>3167</td>
</tr>
<tr>
<td><a href="http://www.wipo.int">www.wipo.int</a></td>
<td>3150</td>
</tr>
<tr>
<td>news.bbc.co.uk</td>
<td>2733</td>
</tr>
<tr>
<td>crl.microsoft.com</td>
<td>2595</td>
</tr>
</tbody>
</table>

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

50395384  n\x00o\x00m\x00b\x00r\x00e\x00\x001\x002\x003\x00@\x00h\x00o
\x00t\x000\x00a\x00i\x00l\x00\x00.\x00c\x00o\x00m\x00\x00 e\x00m\x00p\x001\x000
\x00: \x00\x00A\x00\x09\x00n\x00o\x00m\x00b\x00r\x00e\x00\x001\x002\x003\x00@\x00n
\x00o\x00t\x00m\x00\x00i\x00l\x00\x00.\x00c\x00o\x00m\x00\x00A\x00\x09\x00m\x00i\x00n
\x00o\x00m\x00b\x00

50395432  m\x00i\x00o\x00n\x00o\x00m\x00b\x00r\x00e\x00\x00m\x00s\x00n
\x00.\x00c\x00o\x00m\x00\x00i\x00l\x00\x00.\x00c\x00o\x00m\x00\x00A\x00\x09\x00m
\x00i\x00n\x00o\x00b\x00r\x00e\x00\x00@\x00m\x00s\x00n\x00.\x00c\x00o\x00m
\x00\x00A\x00\x09\x00e\x00j\x00e\x00m\x00p\x001\x000

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

Further down we see:

828564544  charlie@m57.biz  (37190)\x0D\x0A\x09  for <charlie@m57.biz>; Fri, 20 Nov 2

828564992  4B01C378.3060603@m57.biz  0\x0D\x0A References:
<4B01C378.3060603@m57.biz>\x0D\x0ATo: charlie@m

828565023  charlie@m57.biz  3@m57.biz>\x0D\x0ATo: charlie@m57.biz\x0D\x0ASubject: Still
email_histogram.txt shows a histogram of all potential email addresses

Clearly the histogram makes a difference:

# 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: email
# Histogram-File-Version: 1.1
n=875   mozilla@kewis.ch
n=651   charlie@m57.biz
n=605   ajbanck@planet.nl
n=411   mikep@oeone.com
n=395   belhaire@ief.u-psud.fr
n=379   premium-server@thawte.com
n=356   lilmatt@mozilla.com
n=312   cedric.corazza@wanadoo.fr

Notice:

- 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/cqi-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\x00\xF1
\xE5\x01\x00\x15\x02
```

Note that these “domains” are *not* included in the domain histogram!
Analyze the impact of find with the tests/regress.py script

$ python bulk_extractor-1.3/tests/regress.py --ana charlie-2009-12-11-find
Analyze charlie-2009-12-11-find
bulk_extractor version: 1.3b6-dev001
Filename: /corp/nps/scenarios/2009-m57-patents/drives-redacted/
charlie-2009-12-11.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>FIND</td>
<td>2442</td>
<td>18658.7566</td>
<td>7.6408</td>
<td>58.19%</td>
</tr>
<tr>
<td>HIBER</td>
<td>2442</td>
<td>3089.4381</td>
<td>1.2651</td>
<td>9.63%</td>
</tr>
<tr>
<td>HIBER-FIND</td>
<td>19</td>
<td>2528.6812</td>
<td>133.0885</td>
<td>7.89%</td>
</tr>
<tr>
<td>EMAIL</td>
<td>2442</td>
<td>1898.3456</td>
<td>0.7774</td>
<td>5.92%</td>
</tr>
<tr>
<td>ACCTS</td>
<td>2442</td>
<td>1124.7081</td>
<td>0.4606</td>
<td>3.51%</td>
</tr>
<tr>
<td>BASE16</td>
<td>2442</td>
<td>830.0153</td>
<td>0.3399</td>
<td>2.59%</td>
</tr>
<tr>
<td>NET</td>
<td>2442</td>
<td>714.8930</td>
<td>0.2927</td>
<td>2.23%</td>
</tr>
<tr>
<td>ZIP</td>
<td>2442</td>
<td>588.4703</td>
<td>0.2410</td>
<td>1.84%</td>
</tr>
<tr>
<td>AES</td>
<td>2442</td>
<td>588.0705</td>
<td>0.2408</td>
<td>1.83%</td>
</tr>
<tr>
<td>ZIP-FIND</td>
<td>75052</td>
<td>401.8809</td>
<td>0.0054</td>
<td>1.25%</td>
</tr>
<tr>
<td>GZIP</td>
<td>2442</td>
<td>151.1465</td>
<td>0.0619</td>
<td>0.47%</td>
</tr>
<tr>
<td>WINPE</td>
<td>2442</td>
<td>117.5074</td>
<td>0.0481</td>
<td>0.37%</td>
</tr>
<tr>
<td>EXIF</td>
<td>2442</td>
<td>114.9825</td>
<td>0.0471</td>
<td>0.36%</td>
</tr>
<tr>
<td>HIBER-EMAIL</td>
<td>19</td>
<td>104.6778</td>
<td>5.5094</td>
<td>0.33%</td>
</tr>
<tr>
<td>GZIP-FIND</td>
<td>1973</td>
<td>97.8214</td>
<td>0.0496</td>
<td>0.31%</td>
</tr>
<tr>
<td>HIBER-ACCTS</td>
<td>19</td>
<td>91.1740</td>
<td>4.7986</td>
<td>0.28%</td>
</tr>
<tr>
<td>HIBER-AES</td>
<td>19</td>
<td>87.1285</td>
<td>4.5857</td>
<td>0.27%</td>
</tr>
<tr>
<td>JSON</td>
<td>2442</td>
<td>81.9399</td>
<td>0.0336</td>
<td>0.26%</td>
</tr>
<tr>
<td>PDF</td>
<td>2442</td>
<td>77.1401</td>
<td>0.0316</td>
<td>0.24%</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Offset</th>
<th>Array Length</th>
<th>MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5091418457</td>
<td>[6, 4, 6, 4]</td>
<td>7ea5995a7acbd301b98e15b50b723e2b</td>
</tr>
<tr>
<td>5091418525</td>
<td>[6, 4, 6, 4]</td>
<td>7ea5995a7acbd301b98e15b50b723e2b</td>
</tr>
</tbody>
</table>

10002203123  
{"url":"http://patft.uspto.gov/netacgi/nph-Parser?
 Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%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

Good luck with this!

- 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 —
telephone.txt — Phone numbers!

Beware — many are tech support!

<table>
<thead>
<tr>
<th>Phone number</th>
<th>Area Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>88850883</td>
<td>(800)</td>
<td>Tech support information centre: (800) 563-9048</td>
</tr>
<tr>
<td>88850995</td>
<td>(905)</td>
<td>Microsoft: (905) 568-4494</td>
</tr>
<tr>
<td>88851056</td>
<td>(905)</td>
<td>Other stats: (905) 568-2294</td>
</tr>
<tr>
<td>88851111</td>
<td>(905)</td>
<td>Technical support: (905) 568-3503</td>
</tr>
<tr>
<td>88851162</td>
<td>(800)</td>
<td>Priority: (800) 668-7975</td>
</tr>
<tr>
<td>88851208</td>
<td>(905)</td>
<td>Text phone (TTY/TDD): (905) 568-9641</td>
</tr>
<tr>
<td>88851367</td>
<td>(809)</td>
<td>Phone: (809) 273-3600</td>
</tr>
</tbody>
</table>

Some are bogus:

<table>
<thead>
<tr>
<th>Phone number</th>
<th>Area Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5054603274</td>
<td>800-888-8800</td>
<td>Argentina, 0-800-888-8800, 54-11-4317-2626</td>
</tr>
<tr>
<td>5054604226</td>
<td>+27 11 257 0000</td>
<td>not available, +27 11 257 0000, 800-888-8800, 0-800-888-8800, 54-11-4317-2626</td>
</tr>
<tr>
<td>5054604316</td>
<td>800-701-1774</td>
<td>RA, Brazil, 0-800-701-1774, (11) 3328-3396</td>
</tr>
<tr>
<td>5054604402</td>
<td>+27 11 257 0000</td>
<td>not available, +27 11 257 0000, 800-888-8800, 0-800-888-8800, 54-11-4317-2626</td>
</tr>
<tr>
<td>5054604738</td>
<td>888-571-2048</td>
<td>CAN, Canada, 888-571-2048, 716-871-2929, 800-888-8800, 0-800-888-8800, 54-11-4317-2626</td>
</tr>
<tr>
<td>5054604751</td>
<td>716-871-2929</td>
<td>CAN, CAN, 888-571-2048, 716-871-2929, 800-888-8800, 0-800-888-8800, 54-11-4317-2626</td>
</tr>
<tr>
<td>5054605064</td>
<td>800 830-1832</td>
<td>\x0D\x0ACHN, China, 800 830-1832, +86 755 8340 9</td>
</tr>
</tbody>
</table>

And some are clearly legit:

<table>
<thead>
<tr>
<th>Phone number</th>
<th>Area Code</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 phone numbers

| 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%2Fnetacqi%2Fnph-Parser%3FSect1%3DPTO2%252526Sect2%3DHIT center>
\x0A<a href=http://patimg2.uspto.gov/.piw?Docid=07626151&homeurl=http%3A%2F%
%2Fpatft.uspto.gov%2Fnetacqi%2Fnph-Parser%3FSect1%3DPTO2%252526Sect2%3DHITO
FF%252526p %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
\x00N\x00o\x00 \x00e\x00n\x00t\x00a\x00

53952231 http://appft1.uspto.gov/netacqi/nph-Parser?
TERM1=20020186464&Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetacqi%2
FPTO %2Fsrchnum.html&r=0& =left>
\x0A<a href="http://appft1.uspto.gov/netacqi/nph-
Parser?TERM1=20020186464&Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetacqi%2
FPTO %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/netacqi/nph
n=2  http://ebiz1.uspto.gov/vision-service/ShoppingCart_P/AddToShoppingCart?
docNumber=7627056&backUrl1=http%3A//patft1.uspto.gov/netacqi/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.xu  (utf16=858)
url_facebook, 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 no vcard entries in charlie-2009-12-11.
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 AES encryption keys, usually found in RAM, Swap, or hibernation files

bulk_extractor feature files now begin with a BOM and header info:

```
# 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: aes_keys
# Feature-File-Version: 1.1

- Any “scanner” can record in any feature file.

Next comes the data:

```

1023960572 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15
16 17 18 19 1a 1b 1c 1d 1e 1f  AES256

1026549244 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15
16 17 18 19 1a 1b 1c 1d 1e 1f  AES256

- These keys appear to be AES test vectors from a Windows DLL.
- We see them on many disk images.
exif.txt is a list of every EXIF that is found on the drive

This feature file has a different internal formatting:

| offset | MD5 of first 4K of JPEG | XML encoding of EXIF |

These files are *really* hard to understand...

Fortunately, we have a program that turns it into a spreadsheet...
$ 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 GPS info extracted from JPEGs and Garmin XML files.

This is interesting because it’s data from other devices (cameras, etc.)
We honestly don’t know if it is useful to put in times without lat/lon info.

- What do you think?
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?
kml.txt — KML files (typically from Google Maps & Earth)

There is no KML in the M57-Patents corpora
You will find most of the disk entries:

3230706176  EXCH_ntfsdrv.dll  <fileobject src='mft'>
<atime>2009-11-09T01:24:59Z</atime><attr_flags>2080</attr_flags>
<filename>EXCH_ntfsdrv.dll</filename><filesize>38912</filesize>
<filesize_alloc>40960</filesize_alloc><lsn>123102367</lsn>
<mtime>2001-08-18T06:36:28Z</mtime><nlink>2</nlink><par_ref>71</par_ref>
<par_seq>1</par_seq><seq>2</seq></fileobject>

3230707200  ntio404.sys   <fileobject src='mft'>
<atime>2009-11-09T01:24:59Z</atime><attr_flags>2080</attr_flags>
<crt ime>2008-04-14T12:00:00Z</crt ime><ctime>2009-11-08T17:08:04Z</ctime>
<filename>ntio404.sys</filename><filesize>34560</filesize>
<filesize_alloc>65536</filesize_alloc><lsn>29295332</lsn>
<mtime>2008-04-14T12:00:00Z</mtime><nlink>1</nlink><par_ref>71</par_ref>
<par_seq>1</par_seq><seq>1</seq></fileobject>

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

159466528  -eSigPol.icy  <fileobject src='fat'>
<atime>2037-09-13T00:00:00</atime><attrib>45</attrib><ctime>2030-03-09T00:00:00</ctime>
<ctimeten>56</ctimeten><filename>-eSigPol.icy</filename><filesize>1937007917</filesize>
<mtime>2034-09-13T12:43:13</mtime><startcluster>1701667951</startcluster></fileobject>

173063680-GZIP-470016  dukdxd1o.lH7  <fileobject src='fat'>
<atime>2010-09-29T00:00:00</atime><attrib>32</attrib><ctime>1999-09-25T06:34:01</ctime>
<ctimeten>50</ctimeten><filename>dukdx d1o.lH7</filename><filesize>1632198449</filesize>
<mtime>2007-01-18T15:01:17</mtime><start cluster>2016504113</start cluster></fileobject>

Q: should we ignore startcluster > disksize?
winpe.txt — Windows executables (New in BE1.3!)
(Don’t worry — explained on next page!)

117886464  0316eaac06e782616036639824c04ceb <PE>
  <FileHeader Machine="IMAGE_FILE_MACHINE_I386" NumberOfSections="5" TimeDateStamp="1255540604"
    PointerToSymbolTable="0" NumberOfSymbols="0" SizeOfOptionalHeader="224">
  <Characteristics><IMAGE_FILE_EXECUTABLE_IMAGE /></Characteristics></FileHeader>
  <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>
  <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 Name=".rdata" VirtualSize="df22" VirtualAddress="41000"
      SizeOfRawData="e000" PointerToRawData="400" PointerToRelocations="0" PointerToLinenumbers="0"
      Characteristics><IMAGE_SCN_CNT_INITIALIZED_DATA /><IMAGE_SCN_MEM_READ /></Characteristics>
    <SectionHeader Name=".data" VirtualSize="1128" VirtualAddress="4f000"
      SizeOfRawData="a00" PointerToRawData="4dc00" PointerToRelocations="0" PointerToLinenumbers="0"
      Characteristics><IMAGE_SCN_CNT_INITIALIZED_DATA /><IMAGE_SCN_MEM_READ /><IMAGE_SCN_MEM_WRITE /></Characteristics>
    <SectionHeader Name=".rsrc" VirtualSize="848" VirtualAddress="51000"
      SizeOfRawData="a00" PointerToRawData="4e600" PointerToRelocations="0" PointerToLinenumbers="0"
      Characteristics><IMAGE_SCN_CNT_INITIALIZED_DATA /></Characteristics>
    <SectionHeader Name=".reloc" VirtualSize="672c" VirtualAddress="52000"
      SizeOfRawData="6800" PointerToRawData="4f000" PointerToRelocations="0" PointerToLinenumbers="0"
      Characteristics><IMAGE_SCN_CNT_INITIALIZED_DATA /></Characteristics>
    <SectionHeader Name=".text" VirtualSize="1128" VirtualAddress="4f000"
      SizeOfRawData="a00" PointerToRawData="4dc00" PointerToRelocations="0" PointerToLinenumbers="0"
      Characteristics><IMAGE_SCN_CNT_INITIALIZED_DATA /></Characteristics>
  </Sections>
  <dlls>
    <dll>ADVAPI32.dll</dll>
    <dll>WS2_32.dll</dll>
  </dlls>
</PE>
Uses:

- Offset tells you were the 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.
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 header of every zip archive

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 ethernet addresses (from packets and ASCII)

342699417 00:80:77:31:01:07 n008077310107 1 00:80:77:31:01:07 192.168.1.2 an
342700437 00:80:77:31:01:07 the following: 00:80:77:31:01:07 brn008077310107
559251251 00:80:77:31:01:07 N008077310107 1 00:80:77:31:01:07 192.168.1.2</sport>
567912435 00:80:77:31:01:07 class="command">00:80:77:31:01:07 BRN0080773101
684600847 00:80:77:31:01:07 -s 192.168.1.2 00:80:77:31:01:07</span></div><a
6341279242 00:0B:DB:4F:6B:10 (ether_dhost)
6341279242 00:19:E3:E7:5D:23 (ether_shost)
6341283338 00:0B:DB:4F:6B:10 (ether_dhost)
6341283338 00:19:E3:E7:5D:23 (ether_shost)
6341287434 00:0B:DB:4F:6B:10 (ether_dhost)

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: ip addresses from packet carving (scan_net) (not from dotted quads)

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

```plaintext
# Filename: /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01
# Feature-Recorder: ip
# Feature-File-Version: 1.1

117942521       20.137.78.24    struct ip R (src) cksum-bad
117942521       94.89.93.194    struct ip L (dst) cksum-bad
118342942       20.137.78.24    struct ip R (src) cksum-bad
118342942       94.89.93.194    struct ip L (dst) cksum-bad

9977306594      192.168.1.1     sockaddr_in
9977393926      63.245.209.93  sockaddr_in

5839793854-HIBER-17952268       90.4.162.232    struct ip L (dst) cksum-bad
5839793854-HIBER-17960460       78.0.3.185      struct ip R (src) cksum-bad
5839793854-HIBER-17960460       90.4.162.232    struct ip L (dst) cksum-bad
6339825268      192.168.1.104   struct ip L (src) cksum-ok
6339825268      192.168.1.1     struct ip R (dst) cksum-ok
6339825320      192.168.1.104   struct ip L (src) cksum-ok

5839793854-HIBER-129985200      8.3.2.3  sockaddr_in
```
ip_histogram.txt removes random noise
(1.3 histogram is only of chksum-ok values)

Histogram of all values:

```plaintext
# 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 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

- This has a time zone of -0600 (I’m typing this in Utah in the summer)
- The time in the packet file is “1”
  /* Possibly a valid ethernet frame but not preceded 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>Source IP Address</th>
<th>Destination IP Address</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.137.78.24:2048</td>
<td>94.89.93.194:32824</td>
<td></td>
<td></td>
<td>TCP</td>
<td>232</td>
</tr>
<tr>
<td>20.137.78.24:2048</td>
<td>94.89.93.194:32824</td>
<td></td>
<td></td>
<td>TCP</td>
<td>232</td>
</tr>
<tr>
<td>255.144.140.1:3972</td>
<td>0.0.133.192:52224</td>
<td></td>
<td></td>
<td>TCP</td>
<td>3973</td>
</tr>
<tr>
<td>1.0.0.0:0</td>
<td>117.17.2.0:0</td>
<td></td>
<td></td>
<td>UDP</td>
<td>512</td>
</tr>
<tr>
<td>56.141.76.36:65490</td>
<td>28.81.139.206:35832</td>
<td></td>
<td></td>
<td>TCP</td>
<td>3972</td>
</tr>
<tr>
<td>56.141.76.36:65490</td>
<td>28.81.139.206:35832</td>
<td></td>
<td></td>
<td>TCP</td>
<td>3972</td>
</tr>
<tr>
<td>56.141.76.36:65490</td>
<td>28.81.139.206:35832</td>
<td></td>
<td></td>
<td>TCP</td>
<td>3972</td>
</tr>
<tr>
<td>20.137.78.24:2048</td>
<td>94.89.93.194:21899</td>
<td></td>
<td></td>
<td>TCP</td>
<td>232</td>
</tr>
<tr>
<td>7.90.102.193:13311</td>
<td>108.5.218.9:18387</td>
<td></td>
<td></td>
<td>TCP</td>
<td>3973</td>
</tr>
</tbody>
</table>

Be careful of false positives:

<table>
<thead>
<tr>
<th>Source IP Address</th>
<th>Destination IP Address</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0:101</td>
<td>0.0.0.0:19829</td>
<td></td>
<td></td>
<td>TCP</td>
<td>77</td>
</tr>
<tr>
<td>48.144.141.49:0</td>
<td>176.61.0.0:0</td>
<td></td>
<td></td>
<td>TCP</td>
<td>70</td>
</tr>
<tr>
<td>7.86.252.232:55425</td>
<td>47.0.250.69:21841</td>
<td></td>
<td></td>
<td>TCP</td>
<td>1419</td>
</tr>
<tr>
<td>255.118.14.233:5760</td>
<td>255.164.149.80:52428</td>
<td></td>
<td></td>
<td>UDP</td>
<td>768</td>
</tr>
<tr>
<td>255.118.14.233:5760</td>
<td>255.164.149.80:52428</td>
<td></td>
<td></td>
<td>UDP</td>
<td>768</td>
</tr>
<tr>
<td>255.118.14.233:5760</td>
<td>255.164.149.80:52428</td>
<td></td>
<td></td>
<td>UDP</td>
<td>768</td>
</tr>
<tr>
<td>255.118.14.233:5760</td>
<td>255.164.149.80:52428</td>
<td></td>
<td></td>
<td>UDP</td>
<td>768</td>
</tr>
<tr>
<td>57.93.93.93:3968</td>
<td>8.141.88.247:17843</td>
<td></td>
<td></td>
<td>TCP</td>
<td>5631</td>
</tr>
<tr>
<td>57.93.93.93:3968</td>
<td>8.141.88.247:17843</td>
<td></td>
<td></td>
<td>TCP</td>
<td>5631</td>
</tr>
</tbody>
</table>
tcp_histogram.txt — would be nice to have total flow info

These packets:

<table>
<thead>
<tr>
<th></th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>101727492</td>
<td>56.141.76.36:65490</td>
<td>28.81.139.206:35832</td>
<td>TCP</td>
<td>3972</td>
</tr>
<tr>
<td>102361428</td>
<td>56.141.76.36:65490</td>
<td>28.81.139.206:35832</td>
<td>TCP</td>
<td>3972</td>
</tr>
</tbody>
</table>

Become this histogram:

<table>
<thead>
<tr>
<th>n</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>7.90.102.193:13311</td>
<td>108.5.218.9:18387</td>
<td>TCP</td>
<td>128.102.193.13311:128.5.218.9:18387</td>
</tr>
<tr>
<td>53</td>
<td>0.0.123.55:12288</td>
<td>56.49.57.65:12336</td>
<td>TCP</td>
<td>0.0.123.55:12288:123.55:12288:56.49.57.65:12336</td>
</tr>
<tr>
<td>48</td>
<td>5.100.228.83:64</td>
<td>15.134.211.0:15</td>
<td>TCP</td>
<td>5.100.228.83:64:15.134.211.0:15</td>
</tr>
<tr>
<td>38</td>
<td>252.21.212.255:34048</td>
<td>83.0.0.0:17792</td>
<td>TCP</td>
<td>252.21.212.255:34048:83.0.0.0:17792</td>
</tr>
<tr>
<td>38</td>
<td>252.21.212.255:34048</td>
<td>83.0.16.16:17792</td>
<td>TCP</td>
<td>252.21.212.255:34048:83.0.16.16:17792</td>
</tr>
<tr>
<td>30</td>
<td>5.225.0.252:61133</td>
<td>176.69.248.3:63488</td>
<td>TCP</td>
<td>5.225.0.252:61133:176.69.248.3:63488</td>
</tr>
<tr>
<td>28</td>
<td>0.106.37.95:23179</td>
<td>102.59.199.117:52968</td>
<td>TCP</td>
<td>0.106.37.95:23179:102.59.199.117:52968</td>
</tr>
<tr>
<td>27</td>
<td>20.137.78.24:2048</td>
<td>94.89.93.194:21899</td>
<td>TCP</td>
<td>20.137.78.24:2048:94.89.93.194:21899</td>
</tr>
<tr>
<td>24</td>
<td>106.0.80.83:51457</td>
<td>141.74.255.139:65382</td>
<td>UDP</td>
<td>106.0.80.83:51457:141.74.255.139:65382</td>
</tr>
</tbody>
</table>

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 ELF executables

charlie-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" ehsizetime="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></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><SHFALLOC /></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 <ELF> record...

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

```
# Feature-File-Version: 1.1
727114768-GZIP-2048 ...
```

The MD5 is the hash of the first 4KiB:

```
1b5984e4365278bee12c9be8849439f4
```

Next comes the XML for the header:

```
<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" ehsizen"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></flags>
  </section>
  ...
</sections>
<shared_objects><so>libc.so.6</so></shared_objects>
</ELF>
```
<PE> <FileHeader> provides information on header

```xml
<?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>
```
<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>
<PE><Sections><SectionHeader
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>
```
Suppressing bulk_extractor False Positives
For this section we will work with ubnist1

nps-2009-ubnist1 — Bootable Linux USB used to browse USG sites

-rw-rw-r-- 1 simsong staff 685 Jan 29 2009 narrative.txt
-rw-rw-r-- 1 simsong staff 725408916 Jan 6 2009 ubnist1.gen0.aff
-rw-rw-r-- 1 simsong staff 2106589184 Jan 6 2009 ubnist1.gen0.raw
-rw-rw-r-- 1 simsong staff 746023186 Jan 6 2009 ubnist1.gen1.aff
-rw-rw-r-- 1 simsong staff 2106589184 Jan 6 2009 ubnist1.gen1.raw
-rw-rw-r-- 1 simsong staff 838513850 Jan 9 2009 ubnist1.gen2.aff
-rw-rw-r-- 1 simsong staff 2106589184 Jan 9 2009 ubnist1.gen2.raw
-rw-rw-r-- 1 simsong staff 1572852102 May 2 2010 ubnist1.gen3.E01
-rw-rw-r-- 1 simsong staff 534512293 May 2 2010 ubnist1.gen3.E02
-rw-rw-r-- 1 simsong staff 890164681 Jan 7 2009 ubnist1.gen3.aff
-rw-rw-r-- 1 simsong staff 2106589184 Jan 7 2009 ubnist1.gen3.raw
Hard drives are *filled* with email addresses.

Bulk_extractor finds email addresses in many places:
- Windows binaries; SSL certificates
- Documents
- Cached web pages; Memory; Hibernation Files

**UBNIST1** have a LOT of email addresses; each snapshot sees more...

<table>
<thead>
<tr>
<th>gen0</th>
<th>gen1</th>
<th>gen2</th>
<th>gen3</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=3447 <a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>n=3455 <a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>n=27364 <a href="mailto:ubuntu-users@lists.ubuntu.com">ubuntu-users@lists.ubuntu.com</a></td>
<td>n=27672 <a href="mailto:ubuntu-users@lists.ubuntu.com">ubuntu-users@lists.ubuntu.com</a></td>
</tr>
<tr>
<td>n=2237 <a href="mailto:hadess@hadess.net">hadess@hadess.net</a></td>
<td>n=3254 <a href="mailto:ubuntu-devel-discuss@lists.ubuntu.com">ubuntu-devel-discuss@lists.ubuntu.com</a></td>
<td>n=17213 <a href="mailto:ubuntu-motu@lists.ubuntu.com">ubuntu-motu@lists.ubuntu.com</a></td>
<td>n=17133 <a href="mailto:ubuntu-motu@lists.ubuntu.com">ubuntu-motu@lists.ubuntu.com</a></td>
</tr>
<tr>
<td>n=2040 <a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>n=2241 <a href="mailto:hadess@hadess.net">hadess@hadess.net</a></td>
<td>n=14291 <a href="mailto:ubuntu-devel-discuss@lists.ubuntu.com">ubuntu-devel-discuss@lists.ubuntu.com</a></td>
<td>n=12936 <a href="mailto:ubuntu-devel-discuss@lists.ubuntu.com">ubuntu-devel-discuss@lists.ubuntu.com</a></td>
</tr>
<tr>
<td>n=990 <a href="mailto:airlied@linux.ie">airlied@linux.ie</a></td>
<td>n=990 <a href="mailto:airlied@linux.ie">airlied@linux.ie</a></td>
<td>n=4086 <a href="mailto:language-packs@ubuntu.com">language-packs@ubuntu.com</a></td>
<td>n=4032 <a href="mailto:language-packs@ubuntu.com">language-packs@ubuntu.com</a></td>
</tr>
<tr>
<td>n=910 <a href="mailto:cjwatson@debian.org">cjwatson@debian.org</a></td>
<td>n=930 <a href="mailto:cjwatson@debian.org">cjwatson@debian.org</a></td>
<td>n=990 <a href="mailto:rene@debian.org">rene@debian.org</a></td>
<td>n=3481 <a href="mailto:olly@survex.com">olly@survex.com</a></td>
</tr>
<tr>
<td>n=893 <a href="mailto:rene@debian.org">rene@debian.org</a></td>
<td>n=910 <a href="mailto:rene@debian.org">rene@debian.org</a></td>
<td>n=2280 <a href="mailto:ubuntu-desktop@lists.ubuntu.com">ubuntu-desktop@lists.ubuntu.com</a></td>
<td>n=2239 <a href="mailto:hadess@hadess.net">hadess@hadess.net</a></td>
</tr>
<tr>
<td>n=884 <a href="mailto:anholt@freebsd.org">anholt@freebsd.org</a></td>
<td>n=909 <a href="mailto:dsandras@seconix.com">dsandras@seconix.com</a></td>
<td>n=2239 <a href="mailto:hadess@hadess.net">hadess@hadess.net</a></td>
<td>n=2040 <a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
</tr>
<tr>
<td>n=767 <a href="mailto:jirka@5z.com">jirka@5z.com</a></td>
<td>n=767 <a href="mailto:jirka@5z.com">jirka@5z.com</a></td>
<td>n=2040 <a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>n=1966 <a href="mailto:ubuntu-desktop@lists.ubuntu.com">ubuntu-desktop@lists.ubuntu.com</a></td>
</tr>
<tr>
<td>n=760 <a href="mailto:guillem@debian.org">guillem@debian.org</a></td>
<td>n=760 <a href="mailto:guillem@debian.org">guillem@debian.org</a></td>
<td>n=1696 <a href="mailto:debian-x@lists.debian.org">debian-x@lists.debian.org</a></td>
<td>n=1484 <a href="mailto:debian-x@lists.debian.org">debian-x@lists.debian.org</a></td>
</tr>
<tr>
<td>n=744 <a href="mailto:wakkerma@debian.org">wakkerma@debian.org</a></td>
<td>n=744 <a href="mailto:wakkerma@debian.org">wakkerma@debian.org</a></td>
<td>n=1202 <a href="mailto:strk@keybit.net">strk@keybit.net</a></td>
<td>n=1202 <a href="mailto:strk@keybit.net">strk@keybit.net</a></td>
</tr>
<tr>
<td>n=708 <a href="mailto:dshaw@jabberwocky.com">dshaw@jabberwocky.com</a></td>
<td>n=708 <a href="mailto:dshaw@jabberwocky.com">dshaw@jabberwocky.com</a></td>
<td>n=1122 <a href="mailto:bwy@benjaminwolsey.de">bwy@benjaminwolsey.de</a></td>
<td>n=1122 <a href="mailto:bwy@benjaminwolsey.de">bwy@benjaminwolsey.de</a></td>
</tr>
<tr>
<td>n=660 <a href="mailto:doogie@debian.org">doogie@debian.org</a></td>
<td>n=660 <a href="mailto:doogie@debian.org">doogie@debian.org</a></td>
<td>n=1044 <a href="mailto:pkg-perl-maintainers@lists.alioth.debian.org">pkg-perl-maintainers@lists.alioth.debian.org</a></td>
<td>n=1023 <a href="mailto:cjwatson@debian.org">cjwatson@debian.org</a></td>
</tr>
<tr>
<td>n=659 <a href="mailto:brian.cameron@sun.com">brian.cameron@sun.com</a></td>
<td>n=659 <a href="mailto:brian.cameron@sun.com">brian.cameron@sun.com</a></td>
<td>n=1036 <a href="mailto:cjwatson@debian.org">cjwatson@debian.org</a></td>
<td>n=1010 <a href="mailto:rene@debian.org">rene@debian.org</a></td>
</tr>
<tr>
<td>n=656 <a href="mailto:dsandras@gnome.org">dsandras@gnome.org</a></td>
<td>n=656 <a href="mailto:dsandras@gnome.org">dsandras@gnome.org</a></td>
<td>n=1017 <a href="mailto:rene@debian.org">rene@debian.org</a></td>
<td>n=1006 <a href="mailto:pkg-perl-maintainers@lists.alioth.debian.org">pkg-perl-maintainers@lists.alioth.debian.org</a></td>
</tr>
<tr>
<td>n=654 <a href="mailto:kyoshida@novell.com">kyoshida@novell.com</a></td>
<td>n=654 <a href="mailto:kyoshida@novell.com">kyoshida@novell.com</a></td>
<td>n=990 <a href="mailto:airlied@linux.ie">airlied@linux.ie</a></td>
<td>n=990 <a href="mailto:airlied@linux.ie">airlied@linux.ie</a></td>
</tr>
<tr>
<td>n=632 <a href="mailto:priikone@silcnet.org">priikone@silcnet.org</a></td>
<td>n=632 <a href="mailto:priikone@silcnet.org">priikone@silcnet.org</a></td>
<td>n=909 <a href="mailto:dsandras@seconix.com">dsandras@seconix.com</a></td>
<td>n=909 <a href="mailto:dsandras@seconix.com">dsandras@seconix.com</a></td>
</tr>
<tr>
<td>n=626 <a href="mailto:daniel@fooishbar.org">daniel@fooishbar.org</a></td>
<td>n=626 <a href="mailto:daniel@fooishbar.org">daniel@fooishbar.org</a></td>
<td>n=884 <a href="mailto:anholt@freebsd.org">anholt@freebsd.org</a></td>
<td>n=884 <a href="mailto:anholt@freebsd.org">anholt@freebsd.org</a></td>
</tr>
</tbody>
</table>

| 6596 total | 6929 total | 8734 total | 8734 total |
It’s important to distinguish email addresses that are relevant to a case from those that are not.

The top address is olly@survex.com
- We should probably ignore Mr. Betts:

Other sources that we might wish to ignore
- Windows binaries; SSL certificates; Sample documents; News Stories
stop lists specify features to be “stopped”

Stopped features are not ignored.
- Stopped features are moved from email.txt to email_stopped.txt.
- This is important for validation and error-diagnosis.

Stop lists are implemented with the word_and_context_list C++ class.
- “words” — Anything that might be in the “feature” column
  — Email address
  — MD5 hash of the first 4KiB of a file (JPEG)
  — AES key
- “context” — Anything that might be in the “context” column
  — Includes the feature
  — Will suppress a specific instance of a feature
- regular expressions
  — Dramatically slows down the process

bulk_extractor also supports alert lists
- “words” or “context” that should be flagged
stop lists and alert lists are specified with the "-w" and "-r" options.

Usage: bulk_extractor [options] imagefile

... 
-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.)

The list can be a list of words or a feature file

- For example
  — stop_list.txt:
    olly@survex.com
    hadess@hadess.net
    daniel@veillard.com

  — alert_list.txt:
    daniel@fooishbar.org

- command to use is:
  bulk_extractor -r alert_list.txt -w stop_list.txt -o ubnist1.gen0-v1 ubnist1.gen0.raw
Stop lists processing is reflected in the feature files.

No stop list:

```
7324005 Aug  4 10:25 ubnist1.gen0/email.txt
169609 Aug  4 10:25 ubnist1.gen0/email_histogram.txt
```

```
$ wc -l ubnist1.gen0/email*
    72965 ubnist1.gen0/email.txt
    6596 ubnist1.gen0/email_histogram.txt
```

With stop list:

```
6566160 Aug  4 11:20 email.txt
169534 Aug  4 11:20 email_histogram.txt
827940 Aug  4 11:20 email_stopped.txt
```

```
$ wc -l ubnist1.gen0-v1/email*
    65241 ubnist1.gen0-v1/email.txt
    6593 ubnist1.gen0-v1/email_histogram.txt
    7729 ubnist1.gen0-v1/email_stopped.txt
```

```
$ head ubnist1.gen0-v1/email_stopped.txt
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# Filename: /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen0.raw
# Feature-Recorder: email_stopped
# Feature-File-Version: 1.1
317986809-GZIP-47 daniel@veillard.com aniel Veillard <daniel@veillard.com>
317986809-GZIP-209 daniel@veillard.com aniel Veillard <daniel@veillard.com>
317986809-GZIP-635 daniel@veillard.com aniel Veillard <daniel@veillard.com>
```

—stop_list.txt:

```
olly@survex.com
hadess@hadess.net
daniel@veillard.com
```

—alert_list.txt:

```
daniel@fooishbar.org
```
Here is more of the email stopped list:

<table>
<thead>
<tr>
<th>Name</th>
<th>Email Address</th>
<th>Message Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>317986809-GZIP-47</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>configure.</td>
</tr>
<tr>
<td>317986809-GZIP-209</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/xs</td>
</tr>
<tr>
<td>317986809-GZIP-635</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/px</td>
</tr>
<tr>
<td>317986809-GZIP-1211</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>doc/xsltpr</td>
</tr>
<tr>
<td>317986809-GZIP-1581</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>configure.</td>
</tr>
<tr>
<td>317986809-GZIP-1692</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libexslt/d</td>
</tr>
<tr>
<td>317986809-GZIP-1824</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>python/gen</td>
</tr>
<tr>
<td>317986809-GZIP-1943</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/xs</td>
</tr>
<tr>
<td>317986809-GZIP-2085</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/px</td>
</tr>
<tr>
<td>317986809-GZIP-2315</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>doc/xsltpr</td>
</tr>
<tr>
<td>317986809-GZIP-2724</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/px</td>
</tr>
<tr>
<td>317986809-GZIP-2940</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>doc/xsltpr</td>
</tr>
<tr>
<td>317986809-GZIP-3331</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>libxslt/px</td>
</tr>
<tr>
<td>317986809-GZIP-3494</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>python/tes</td>
</tr>
<tr>
<td>317986809-GZIP-3647</td>
<td><a href="mailto:daniel@veillard.com">daniel@veillard.com</a></td>
<td>doc/xslt.h</td>
</tr>
<tr>
<td>330031689-GZIP-275885</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>queryparse</td>
</tr>
<tr>
<td>330031689-GZIP-276902</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>tests/api_</td>
</tr>
<tr>
<td>330031689-GZIP-277660</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>configure.</td>
</tr>
<tr>
<td>330031689-GZIP-277778</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>tests/harn</td>
</tr>
<tr>
<td>330031689-GZIP-277951</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>tests/harn</td>
</tr>
<tr>
<td>330031689-GZIP-28061</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>net/remote</td>
</tr>
<tr>
<td>330031689-GZIP-28198</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>tests/harn</td>
</tr>
<tr>
<td>330031689-GZIP-28379</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>tests/harn</td>
</tr>
<tr>
<td>330031689-GZIP-28529</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>configure.</td>
</tr>
<tr>
<td>330031689-GZIP-28655</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>net/progcl</td>
</tr>
<tr>
<td>330031689-GZIP-28783</td>
<td><a href="mailto:olly@survex.com">olly@survex.com</a></td>
<td>net/remote</td>
</tr>
</tbody>
</table>
A feature file can be used as a context stop list. So we can use the gen0 email as a stop for gen1.

This makes the gen0 feature file a *filter*.

This isn’t hard to do in practice:

```
$ src/bulk_extractor -w ubnist1.gen0/email.txt \ 
   -o ubnist1.gen1-filtered_by_gen0 ubnist1.gen1.raw
Reading context stop list ubnist1.gen0/email.txt
Stop list read.
   Total features read: 72961
   List Size: 32084
   Context Strings: 32083
   Regular Expressions: 0
```
The result of filtering gen1 with gen0: We only see the new email addresses

8424165 Aug 4 10:24 ubnist1.gen1/email.txt
179549 Aug 4 10:24 ubnist1.gen1/email_histogram.txt
7324005 Aug 4 10:25 ubnist1.gen0/email.txt
169609 Aug 4 10:25 ubnist1.gen0/email_histogram.txt
1083873 Aug 4 11:49 ubnist1.gen1-filtered_by_gen0/email.txt
13867 Aug 4 11:50 ubnist1.gen1-filtered_by_gen0/email_histogram.txt
7805523 Aug 4 11:49 ubnist1.gen1-filtered_by_gen0/email_stopped.txt

The histogram is a histogram of the filtered email.txt:

n=3178 ubuntu-devel-discuss@lists.ubuntu.com
n=638 ubuntu-desktop@lists.ubuntu.com
n=444 debian-x@lists.debian.org
n=219 debian-boot@lists.debian.org
n=204 ubuntu-motu@lists.ubuntu.com
n=144 seb128@debian.org
n=132 pkg-gnome-maintainers@lists.alioth.debian.org
Stop lists and alert lists have minor impact on performance.

Longer stop lists are slower to process. $t \propto (\text{features} \times \text{stops})$

<table>
<thead>
<tr>
<th>lines in stop list</th>
<th>ubnist1.gen0 Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>54.51 s</td>
</tr>
<tr>
<td>3</td>
<td>55.47 s</td>
</tr>
<tr>
<td>235886</td>
<td>55.41 s</td>
</tr>
</tbody>
</table>

Regular expressions slow down stop lists dramatically:

olly@survex.com
haddess@haddess.net
daniel@veillard.com

.*@survex.com
.*@haddess.net
.*@veillard.com

<table>
<thead>
<tr>
<th>re lines in stop list</th>
<th>ubnist1.gen0 Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>54.51 s</td>
</tr>
<tr>
<td>3</td>
<td>66.41 s</td>
</tr>
</tbody>
</table>

Stop and alert lists must be applied when bulk_extractor is run.

- A future version may allow filtering after-the-fact.
Context-sensitive stop lists are important when looking for unknown individuals.

Recall all of those email addresses on ubnist1

Although these emails are widely seen, they should not be whitelisted:

- Email addresses can be shared
- Email addresses can be sold
- A Linux developer might be engaged in a criminal enterprise

By using context-sensitive stop lists, we:

- Ignore the email address where it is widely seen
- Will still notice the email address in a new context

Context-sensitive lists need to be maintained!

- Build from default installs of operating systems & applications.
- NIST is running bulk_extractor over the entire NSRL and will make the results available.
- Organizations are free to trade the feature files amongst themselves.
post-processing

bulk_diff
identify_filenames.py
cross drive analysis
bulk_diff.py: compare two different bulk_extractor reports

The “report” directory contains:
- DFXML file of bulk_extractor run information
- Multiple feature files

bulk_diff.py: create a “difference report” of two bulk_extractor runs.
- Designed for timeline analysis
- Developed with analysts
- Reports biggest changes at top.
  — Reporting “what’s new” turned out to be more useful
  — “what’s missing” includes data inadvertently overwritten

$ python3.2 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.

Options:
- -h, --help show this help message and exit
- --smaller Also show values that didn't change or got smaller
- --tabdel=TABDEL Specify a tab-delimited output file for easy import into Excel
- --html=HTML HTML output. Argument is file name base
$python3.2 python/bulk_diff.py ubnist1.gen2 ubnist1.gen3
bulk_diff.py Version: 1.0
PRE Image: /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen2.raw
processing ccn_histogram.txt:
No differences

processing ccn_track2_histogram.txt:
No differences

processing domain_histogram.txt:
domain_histogram.txt:
# in PRE # in POST  Δ Value
------------------------------------------------------------------------
  0      5,546 5,546 schemas.openxmlformats.org
  0       569   569 www.nlrb.gov
 705     1,273   568 ns.adobe.com
  0       446   446 media.newjobs.com
  0       404   404 schemas.microsoft.com
...

processing email_histogram.txt:
email_histogram.txt:
# in PRE # in POST  Δ Value
------------------------------------------------------------------------
 27,364    27,672 308 ubuntu-users@lists.ubuntu.com
  314       334  20 kernel-team@lists.ubuntu.com
   88       103  15 akira@debian.org
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) — 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!

Currently the DFXML file has to be reprocessed for each feature file...

$ python3.2 identify_filenames.py --list
charlie-2009-12-11.zip
Feature files in /Users/simsong/charlie-2009-12-11.zip:
ccn.txt
exif.txt
url.txt
url_searches.txt
url_services.txt
ether.txt
domain.txt
windirs.txt
email.txt
ip.txt
aes_keys.txt
zip.txt
rfc822.txt
json.txt
tcp.txt
winpe.txt
gps.txt
winprefetch.txt
telephone.txt
$
identify_filenames can take a long time

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 the charlie-2009-12-10 disk image.
Reviewing the output...

$ 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
$
identify_output feature files have a 4th and 5th column... 
... but only if a file is actually identified...

<table>
<thead>
<tr>
<th>#</th>
<th>Position</th>
<th>Feature</th>
<th>Context</th>
<th>Filename</th>
<th>File MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7277995794</td>
<td>4857994530998756</td>
<td>ible-price/&amp;rnd=4857994530998756\x00request-method\x00</td>
<td>Documents and Settings/Charlie/Local Settings/Application Data/Mozilla/Firefox/Profiles/2usvf7i1.default/Cache/<em>CACHE_001</em></td>
<td>eca068c08645e300edd7530362d80a97</td>
</tr>
</tbody>
</table>

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

3598712863-ZIP-100622 michael.buettner@sun.com chael Büttner
<michael.buettner@sun.com>\x0A - Philipp Documents and Settings/Charlie/My Documents/Downloads/lightning-0.9-tb-win.xpi 70eebfacfe1227e50db99556cf98161e

- position: 3598712863-ZIP-100622
- Feature: michael.buettner@sun.com
- Context: chael Büttner <michael.buettner@sun.com>\x0A - Philip
- Filename: Documents and Settings/Charlie/My Documents/Downloads/lightning-0.9-tb-win.xpi
- File MD5: 70eebfacfe1227e50db99556cf98161e
Never pass up an opportunity to Google an MD5

3598712863-ZIP-100622  michael.buettner@sun.com  chael Büttner
<michael.buettner@sun.com>\x0A - Philipp  Documents and Settings/Charlie/
My Documents/Downloads/lightning-0.9-tb-win.xpi  70eebfacfe1227e50db99556cf98161e

- File MD5: 70eebfacfe1227e50db99556cf98161e

8086ee725f2d3eca17c375a3812c3618  lightning-0.9rc2.linux-i686.xpi
13f72810f33a9817e832e4d929dd1e68  lightning-0.9rc2.mac.xpi
9ec2af4662146905d98d5481f233f86b  lightning-0.9rc2.solaris-i386.xpi
7cd40514e25c476f0fd90e439062b4bf  lightning-0.9rc2.solaris-sparc.xpi
70eebfacfe1227e50db99556cf98161e  lightning-0.9rc2.win32.xpi
identify_filenames.py — Lessons learned

feature file stability:

- Originally feature files had three fields: offset, feature, context
- with identify_filenames, they have four: offset, feature, context, filename
- Other options:
  — *The offset could have been replaced with the filename - but what about features w/o?*
  — *Perhaps each line should be an XML block? Harder to process?*
  — *Perhaps each line should be a JSON object? Harder to protect?*

ASCII vs. UTF-8

- Moving to Python3.2 and UTF-8 forced us to address UNICODE issues throughout BE
- It’s worth the effort: Accents, Arabic, Hebrew in features can be seen natively.

Memory and Algorithm:

- Memory $\propto$ #features ; time $\propto$ #files; charlie-2009-12-10 took 300MiB and
- Originally used linear search through list of extents — Took days to process some images
- Replaced with call to python bisect module — Now took minutes
- Would take less memory by rewriting in C++; faster by making multi-threaded.
**IP Carving and Network Reassembly plug-in**

**bulk_extractor** extended to recognize and validate network data.
- Automated extraction of Ethernet MAC addresses from *IP packets in hibernation files*.

We then re-create the physical networks the computers were on:
Extending bulk_extractor with Plug-ins.
Plugins written in C++

Plugins are distributed as *shared libraries*.
- Windows: `scan_bulk.DLL`
- Mac & Linux: `scan_bulk.so`

Plugins must support a single function call:

```cpp
void scan_bulk(const class scanner_params &sp,
               const recursion_control_block &rcb)
```

- `scanner_params` — Describes what the scanner should do.
  - `sp.sbuf` — SBUF to scan
  - `sp.fs` — Feature recording set to use
    - `sp.phase==0` — initialize
    - `sp.phase==1` — scan the SBUF in `sp.sbuf`
    - `sp.phase==2` — shut down
- `recursion_control_block` — Provides information for recursive calls.

The same plug in system will be used by a future version of tcpflow
Where do we go from here?
bulk_extractor: current status and future goals

Scanners:
- accts  exif  hiberfile  pdf  windir
- aes  find  httpheader  vcard
- base64  gps  json  winprefetch
- ccns  gzip  kml  wordlist
- email headers  net  net  zip

Future Releases:
- bulk (detects encrypted data)
- RAR, RAR2
- LZMA
- BZIP
- NTFS
There are many important areas for research

Algorithm development.
- Adopting to different kinds of data.
- Different resolutions
- Higher Amounts (40TB—40PB)

Software that can…
- Automatically identify outliers and inconsistencies.
- Automatically present complex results in simple, straightforward reports.
- Combine stored data, network data, and Internet-based information.

Many of the techniques here are also applicable to:
- Social Network Analysis.
- Personal Information Management.
- Data mining unstructured information.
Our challenges: innovation, scale & community

Most innovative forensic tools **fail when they are deployed.**

- Production data *much larger* than test data.
  — *One drive might have 10,000 email addresses, another might have 2,000,000.*
- Production data *more heterogeneous* than test data.
- Analysts have less experience & time than tool developers.

How to address?

- Attention to usability & recovery.
- High Performance Computing for testing.
- Programming languages that are *safe* and *high-performance.*

Moving research results from lab to field is itself a research problem.
In summary, there is an urgent need for fundamental research in automated computer forensics. Most work to date has been data recovery and reverse engineering.

- User-level file systems
- Recovery of deleted files.

To solve tomorrow’s hard problems, we need:

- Algorithms that exploit large data sets (>10TB)
- Machine learning to find outliers and inconsistencies.
- Algorithms tolerant of data that is dirty and damaged.

Work in automated forensics is inherently interdisciplinary.

- Systems, Security, and Network Engineering
- Machine Learning
- Natural Language Processing
- Algorithms (compression, decompression, big data)
- High Performance Computing
- Human Computer Interactions