Forensics Visualizations with Open Source Tools

Garfinkel, Simson L.
Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/44318
Forensics Visualizations with Open Source Tools

Simson L. Garfinkel, Ph.D
Associate Professor, Naval Postgraduate School
http://www.simson.net/

Tuesday, November 5th, 2013, 8:40am - 9:15am
Forensic visualizations serve two purposes: Presentation & Discoverery

Presentation — visualizations can explain data

- Report or Courtroom
- Summarize data
- Present time series information — illustrate a sequence of events
- Provide iconic representation of an idea

— *This seems to be what most forensic visualizations are used for*

Discovery — visualizations that help us learn something

- “Situational awareness”
- Network connections
- Summarization
- Correlation

— *This is what we would like to use forensic visualizations for.*
Presentation visualizations: Illustrate complex concepts or sequences of events

DFRWS 2011 Solution, FoxIT

- Data sent between multiple users with multiple devices

These visualizations are typically made by hand.
I use this visualization to explain JPEG segments.

```
$ xxd 1pixel.jpg
0000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0048
0000010: 0048 0000 ffdb 0043 0003 0202 0302 0203
0000020: 0303 0304 0303 0405 0805 0504 0405 0a07
0000030: 0706 080c 0a0c 0c0b 0a0b 0b0d 0e12 100d
0000040: 0e11 0e0b 0b10 1610 1113 1415 1515 0c0f
0000050: 1718 1614 1812 1415 14ff c000 0b08 0001
0000060: 0011 0101 1100 ffc4 01f 0000 0105 0101
0000070: 0101 0101 0000 0000 0000 0102 0304
0000080: 0506 0708 090a 0bff c400 b510 0002 0103
0000090: 0302 0403 0505 0404 0000 017d 0102 0300
00000a0: 0411 0512 2131 4106 1351 6107 2271 1432
00000b0: 8191 a108 2342 b1c1 1552 d1f0 2433 6272
00000c0: 8209 0a16 1718 191a 2526 2728 292a 3435
00000d0: 3637 3839 3a43 4445 4647 4849 4a53 5455
00000e0: 5657 5859 5a63 6465 6667 6869 6a73 7475
00000f0: 7677 7879 7a83 8485 8687 8889 8a92 9394
0000100: 9596 9798 999a a2a3 a4a5 a6a7 a8a9 aab2
0000110: b3b4 b5b6 b7b8 b9ba c2c3 c4c5 c6c7 c8c9
0000120: cad2 d3d4 d5d6 d7d8 d9da e1e2 e3e4 e5e6
0000130: e7e8 e9ea f1f2 f3f4 f5f6 f7f8 f9fa ffda
0000140: 0008 0101 0000 3f00 fd53 afff d9
```

(re-implemented with CSS and HTML)
Explanatory visualizations can have a mix of data-driven and hand-drafted elements.

JPEG HEADER @ byte 0

IN JPEG

Bytes: 31,046

Manual Annotation with Apple Keynote

automatic layout with matplotlib

Sectors: 61
Visualizations do not need to be graphical!
Common visualizations are tables, text files, and hex dumps

Great for discovery... but not very exciting
Instant situational awareness—fast and easy to interpret.
Graphical visualizations are a great way to show geospatial information.

Garfinkel 2005
Omnigraffle

“Hand-drafted”

Garmin GPS

“Data-driven”
Data-driven visualizations are great for showing graphs.

Here we used the graph to test a hypothesis

We thought that we could identify groups of users by correlating MAC addresses found on hard drives.

We had a table, but it was much easier to understand once we drew the graph.

But does this scale?
We can build data-driven visualizations with JavaScript kits. e.g. http://d3js.org/

Methodology:
• Create HTML body, CSS style & JSON data model
• JavaScript reads JSON and creates SVG data elements
• Layout engine makes everything look good

Advantages:
• Everybody has a browser
• Browsers do layout, fonts, etc.
• CSS offers a lot of flexibility
• JavaScript engines can handle big datasets

Issues:
• Requires HTML & CSS design
• Heavy use of JQuery
• Must “fix” output for forensic use
Data-Driven documents should allow for discovery! (e.g. http://linkedjazz.org/network/)
Many of these documents rely on interactivity.
But if you move the mouse, you get a different result.
We are trying to create data-driven visualizations for forensic discovery.

Spot data / trends that were not obvious

Detect clusters & outliers
Visualizations for forensic discovery should be “automatic.”

Data driven

- You create the code — ONCE
- Different data → different graphics

Fixed, predictable, static output:

- Interactive visualizations aren’t appropriate for court
- Different analysts should produce the same visualization

What about interactivity?

- Use interactivity for finding settings
- Use the settings for producing the visualization

What about video?

- Video doesn’t work well in most reports. Is it needed?
This presentation is about making static visualizations of computer forensic data with open source tools.

Why we want static visualization

Issues to consider when making visualizations

How we created a visualization for tcpflow
This data-driven graph shows incidence of credit card numbers on a collection of hard drives.

The graph demonstrates:
- outliers
- total vs. unique

X axis is time of acquisition
It’s not meaningful...

—Data source: strings(1) | ccn_detector | python_scripts

The same graph, annotated based on interviews.

- Data Sources: (previous PDF) + phone calls
Today this is an easier graph to make.

Steps:
1. Run bulk_extractor on disk drives
2. Read the (# unique) and (# total) CCNs in the feature & histogram file
3. Create a bar graph with matplotlib

We ran bulk_extractor on a few thousand drives.

Results were stored in .zip files:

```
/corp/nus/drives_bulk_extractor-2013-08-20:
total used in directory 90286552 available 5713959760
-dwrxr-xr-x. 2 simsong root  61440 Aug  20 14:41 .
-dwrxr-xr-x. 14 corpus irb  4096 Sep  30 09:03 ..
-rw-rw-r--.  1 simsong simsong  31513 Jul  30 11:07 0305.zip
-rw-rw-r--.  1 simsong simsong 1835403 Jul  30 11:07 0308.zip
-rw-rw-r--.  1 simsong simsong  833418 Jul  30 11:07 0310.zip
-rw-rw-r--.  1 simsong simsong  20005 Jul  30 11:07 0312.zip
-rw-rw-r--.  1 simsong simsong 15921378 Jul  30 11:07 0313.zip
-rw-rw-r--.  1 simsong simsong  2938098 Jul  30 11:07 0314.zip
-rw-rw-r--.  1 simsong simsong  3547615 Jul  30 11:07 0315.zip
-rw-rw-r--.  1 simsong simsong  16468 Jul  30 11:07 0498.zip
-rw-rw-r--.  1 simsong simsong  11287 Jul  30 11:06 0572.zip
-rw-rw-r--.  1 simsong simsong  11235 Jul  30 11:07 0574.zip
```
Each ZIP file contains carved objects and feature files:

<table>
<thead>
<tr>
<th>Length</th>
<th>Date</th>
<th>Time</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28-Jul-2013</td>
<td>10:12:10</td>
<td>0313/</td>
</tr>
<tr>
<td>3064</td>
<td>28-Jul-2013</td>
<td>10:09:06</td>
<td>0313/ccn.txt</td>
</tr>
<tr>
<td>435</td>
<td>28-Jul-2013</td>
<td>10:12:02</td>
<td>0313/ccn_histogram.txt</td>
</tr>
<tr>
<td>19440204</td>
<td>28-Jul-2013</td>
<td>10:11:36</td>
<td>0313/domain.txt</td>
</tr>
<tr>
<td>47932</td>
<td>28-Jul-2013</td>
<td>10:12:02</td>
<td>0313/email_histogram.txt</td>
</tr>
<tr>
<td>2291</td>
<td>28-Jul-2013</td>
<td>10:01:44</td>
<td>0313/ether.txt</td>
</tr>
<tr>
<td>0</td>
<td>28-Jul-2013</td>
<td>08:50:02</td>
<td>0313/find.txt</td>
</tr>
<tr>
<td>1675013</td>
<td>28-Jul-2013</td>
<td>10:11:22</td>
<td>0313/jpeg.txt</td>
</tr>
<tr>
<td>0</td>
<td>28-Jul-2013</td>
<td>10:06:24</td>
<td>0313/jpeg/</td>
</tr>
<tr>
<td>27702</td>
<td>28-Jul-2013</td>
<td>10:01:06</td>
<td>0313/jpeg/17849683887-ZIP-0.jpg</td>
</tr>
<tr>
<td>18031</td>
<td>28-Jul-2013</td>
<td>10:01:06</td>
<td>0313/jpeg/17849727138-ZIP-0.jpg</td>
</tr>
<tr>
<td>13734</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17853205179-ZIP-0.jpg</td>
</tr>
<tr>
<td>7393</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17853264313-ZIP-0.jpg</td>
</tr>
<tr>
<td>7805</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17853270606-ZIP-0.jpg</td>
</tr>
<tr>
<td>8799</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17853290623-ZIP-0.jpg</td>
</tr>
<tr>
<td>9358</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17853298338-ZIP-0.jpg</td>
</tr>
<tr>
<td>7446</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17855329427-ZIP-0.jpg</td>
</tr>
<tr>
<td>7267</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17855335786-ZIP-0.jpg</td>
</tr>
<tr>
<td>8407</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17855341964-ZIP-0.jpg</td>
</tr>
<tr>
<td>8951</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17855349323-ZIP-0.jpg</td>
</tr>
<tr>
<td>9861</td>
<td>28-Jul-2013</td>
<td>10:01:12</td>
<td>0313/jpeg/17855366469-ZIP-0.jpg</td>
</tr>
<tr>
<td>10079</td>
<td>28-Jul-2013</td>
<td>10:06:24</td>
<td>0313/jpeg/19117936375-ZIP-0.jpg</td>
</tr>
<tr>
<td>18488</td>
<td>28-Jul-2013</td>
<td>10:10:48</td>
<td>0313/json.txt</td>
</tr>
<tr>
<td>1155</td>
<td>28-Jul-2013</td>
<td>10:06:46</td>
<td>0313/rar.txt</td>
</tr>
<tr>
<td>11166</td>
<td>28-Jul-2013</td>
<td>10:12:02</td>
<td>0313/telephone_histogram.txt</td>
</tr>
<tr>
<td>0</td>
<td>28-Jul-2013</td>
<td>10:12:08</td>
<td>0313/url_facebook-address.txt</td>
</tr>
<tr>
<td>27521</td>
<td>28-Jul-2013</td>
<td>10:12:10</td>
<td>0313/url_searches.txt</td>
</tr>
<tr>
<td>230917</td>
<td>28-Jul-2013</td>
<td>10:12:08</td>
<td>0313/url_services.txt</td>
</tr>
<tr>
<td>0</td>
<td>28-Jul-2013</td>
<td>08:50:02</td>
<td>0313/vcard.txt</td>
</tr>
<tr>
<td>11047565</td>
<td>28-Jul-2013</td>
<td>10:11:40</td>
<td>0313/windirs.txt</td>
</tr>
</tbody>
</table>
# lines in feature file = # of total CCNs
# lines in histogram file = # of “distinct” CCNs

Feature file:

```
.. 15759011793 4874625535450448 2;sz=120x60;ord=4874625535450448\x00\x0D
15759013329 4874625535450448 3;sz=120x60;ord=4874625535450448\x00\x0D
15768826985 4763476767365456 7654'6767676767'4763476767365456?\x01676
..  
```

These are not real CCNs (false positives)

Histogram file:

```
n=10  4874625535450448
n=3   4763476767365456
n=3   5674326276767632
n=2   4228665330004449
n=2   6261031142333000
n=2   6577383247385461
n=1   4353245246356352
n=1   4980541652764985
n=1   5566667778889999
n=1   6444333565233521
n=1   SSN: 666-66-6666
```

We store this in an SQL database:

```
CREATE TABLE files (fileid INTEGER PRIMARY KEY ASC,report_filename TEXT UNIQUE,image_filename TEXT UNIQUE);
CREATE TABLE features (featureid INTEGER PRIMARY KEY ASC,featurename TEXT UNIQUE);
CREATE TABLE counts (countid INTEGER PRIMARY KEY ASC,fileid INTEGER,featureid INTEGER,count INTEGER,
FOREIGN KEY (fileid) references files(fileid),
FOREIGN KEY (featureid) references features(featureid));
CREATE UNIQUE INDEX counts_idx ON counts (fileid,featureid);
```

It's easier to work with data in a database
matplotlib is a python library for making visualizations.

- Python 2 & 3 support
- Multiple output formats
- Integrates with pylab and IPython Notebook
Start by looking at the matplotlib gallery for a similar graph.

pylab_examples example code: bar_stacked.py

(Source code, png, hires.png, pdf)
With the data in a DB, extracting the data we want is easy.

```sql
select image_filename, featurename, count from counts
  natural left join files
  natural left join features
where featurename in ('ccn.txt', 'ccn_histogram.txt')
order by 1;
```

- Produces:
  - AE10-001.E01|ccn_histogram.txt|0
  - AE10-0010.E01|ccn.txt|93
  - AE10-0010.E01|ccn_histogram.txt|18
  - AE10-0011.E01|ccn_histogram.txt|0
  - AE10-0012.E01|ccn_histogram.txt|0
  - AE10-0013.E01|ccn.txt|3
  - AE10-0013.E01|ccn_histogram.txt|1
  - AE10-0014.E01|ccn_histogram.txt|0
```python
def ccngraph(count):
    import numpy as np
    import matplotlib.pyplot as plt

    c = conn.cursor()
    c.execute("select image_filename,count from counts natural left join files natural left join features where featurename='ccn.txt'\n    totals = dict(c.fetchall())

    c.execute("select image_filename,count from counts natural left join files natural left join features where featurename='ccn_histogram.txt'\n    distinct = dict(c.fetchall())

    keys = sorted(list(set(list(totals.keys()) + list(distinct.keys()))))
    names = []
    distinctCounts = []
    totalCounts = []

    for k in keys[0:count]:
        names += [os.path.basename(k)]
        distinctCounts += [distinct.get(k, 0)]
        totalCounts += [totals.get(k, 0)]

    ind = np.arange(count) # the x locations for the groups
    width = 1.0 # the width of the bars: can also be len(x) sequence

    p1 = plt.bar(ind, distinctCounts, width, color='r')
    p2 = plt.bar(ind, totalCounts, width, color='y',bottom=distinctCounts)

    plt.ylabel('# CCNs')
    plt.title('Number of CCNs per drive')
    plt.xticks(ind+width/2., names)
    plt.yticks(np.arange(0,81,10))
    plt.legend((p1[0], p2[0]), ('Distinct', 'Total'))
    plt.show()
```
The result is one HUGE bar and lots of little ones. (First 50)
Viewing first 100, there is another large bar.
Figure 1

Number of CCNs per drive

- Distinct
- Total
First 500...
First 1000...
Today’s graph doesn’t look as good... why?

Key differences:

- Split axes — good for scale
  - *c.f.* logarithmic
  - *150 drives* — easier to read
- Data range 0-32,000
- 5 hours of work
- ≈50 lines of code

- 1000 drives
- Much larger data range (how much?)
- 20 min of work
Fix your graph one issue at a time.

Add the Y units by commenting out `plt.yticks()`:

```python
# plt.yticks(np.arange(0, 81, 10))
```

Data range is 0-340,000
A semi-log plot does a better job showing the range.

Change this:

```python
p1 = plt.bar(ind, distinctCounts, width, color='r')
p2 = plt.bar(ind, totalCounts, width, color='y', bottom=distinctCounts)
```

To this:

```python
# for log plot, never go down to 0
bottom = [.01]*count
p1 = plt.bar(ind, distinctCounts, width, color='r', bottom=bottom)
p2 = plt.bar(ind, totalCounts, width, color='y', bottom=distinctCounts)
plt.yscale('log')
```
The original plot had a “broken axis.”

Use Google to find an example.

![Google search for matplotlib broken y axis example](https://www.google.com/search?q=matplotlib+broken+y+axis)
I quickly found an online example.

Matplotlib - Broken axis example: uneven subplot size

I haven't found a solution to adjust the height of the bottom and top plot of the broken axis example of matplotlib.

BTW: The space between the two plots can be adjusted by:

```python
plt.subplots_adjust(hspace=0.03)
```

UPDATE:

I've almost figured it out using gridspec:

```python
# Broken axis example, where the y-axis will have a portion cut out.
import matplotlib.pyplot as plt
import numpy as np

pts = np.array([0.015, 0.166, 0.133, 0.159, 0.041, 0.024, 0.195, 0.039, 0.161, 0.018, 0.143, 0.056, 0.125, 0.086, 0.094, 0.051, 0.043, 0.021, 0.138, 0.075, 0.109, 0.195, 0.05, 0.074, 0.079, 0.155, 0.02, 0.01, 0.061, 0.008])

pts[[3, 14]] += .8
```
The broken axis is two plots with the same data and different zooms.
Create two plots:

```python
f,(ax,ax2) = plt.subplots(2,1,sharex=True)
..
ax.bar(ind, distinctCounts, width, color='r',bottom=bottom)
ax.bar(ind, totalCounts, width, color='y',bottom=distinctCounts)
..
ax2.bar(ind, distinctCounts, width, color='r',bottom=bottom)
ax2.bar(ind, totalCounts, width, color='y',bottom=distinctCounts)
...
```
Still need to do:

- Add commas to Y axis formatter
- Draw y grid lines across page
- Annotate exciting marks
Output Formats

- PNG
- WMF
- PDF
- JPEG
- SVG
- MP4
Visualization engines support multiple output formats

Bitmaps:
- GIF — Graphic Interchange Format
- PNG — Portable Network Graphics
- JPEG — Joint Photographic Experts Group
  —*Don’t output to bitmaps if you can help it*
  —*Problems with zooming & blurring*

Line art:
- SVG — Scalable Vector Graphics
- PDF — Portable Document Format

Animation:
- MOV — QuickTime
- SWF — Adobe Flash
PDF is a container format. It can distribute a single image or multiple pages.

The FoxIT illustration was extracted from the DFRWS 2011 PDF:
1690681 DFRWS2011_Forensic_Challenge-exported2.pdf

The relevant image is on p. 21
PDF content can be line art or bitmaps.

The FoxIT illustration is a bitmap. Extract it with `pdfimages`:

```bash
$ pdfimages DFRWS2011_Forensic_Challenge-exported2.pdf -f 21 -l 21 -j foxit
$ BLOCKSIZE=1024 ls -s1
  total 16832
    1652 DFRWS2011_Forensic_Challenge-exported2.pdf
    4  foxit-000.jpg
    40 foxit-001.ppm
    4  foxit-002.pbm
    15132 foxit-003.ppm
$ convert foxit-003.ppm foxit-003.png
$ BLOCKSIZE=1024 ls -s1 foxit-003.png
  264 foxit-003.png
$```

267 KB bitmap
Extracted with pdfimages
converted with ImageMagick
Zoomed in, we still have a bitmap. Bitmaps are not “accessible” and can’t be searched.
Many of the JavaScript libraries produce SVG output
SVG can be transformed to PDF with the browser’s “print” command.
The resulting PDF is 2.6MB. You can zoom and search for text.
TCPFLOW 1.4.0
Input: /corp/nps/packets/2008-nitroba/nitroba.pcap
Generated: 2013-10-06 21:20:44
Packets analyzed: 91,144 (55.02 MB)
Transports: IPv4 100%

Top Source Addresses
1) 192.168.8.0/21 - 6.56 MB (11%)
2) 208.111.148.6 - 5.41 MB (9%)
3) 74.125.0.0/20 - 4.55 MB (8%)

Top Destination Addresses
1) 192.168.8.0/21 - 39.58 MB (71%)
2) 192.168.1.64 - 7.77 MB (14%)
3) 239.255.255.250 - 818.61 KB (1%)

Top Source Ports
1) 80 - 44.78 MB (84%)
2) 443 - 1.82 MB (3%)
3) 39710 - 178.85 KB (0%)

Top Destination Ports
1) 80 - 5.85 MB (11%)
2) 39710 - 5.41 MB (10%)
3) 33148 - 4.44 MB (8%)
netviz is a network visualization that we added to tcpflow

Design goals:
• Handle any number of packets
• Output in PDF
• Easy to use without training
• Use the BE13 API

Input: 1 or more PCAP files
• 1–1G packets
• 1–1G connections
• 1–4Gi hosts

Output: PDF file of ≈ 50KB
Netviz is implemented with open source libraries.

<table>
<thead>
<tr>
<th>Output Format</th>
<th>PDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Engine</td>
<td>Cairo</td>
</tr>
<tr>
<td>Layout and Typography</td>
<td>Custom (mistake?)</td>
</tr>
<tr>
<td>Implementation</td>
<td>C++</td>
</tr>
</tbody>
</table>

Other options we considered:

- LaTeX
- HTML & SVG
- HTML5 (JavaScript & Canvas)

We were trying to minimize dependencies
Infoblock provides important forensic information.

Always label your visualizations with:

- Input
- Date of input file
- Date visualization was run
- Command line used to generate the output (we forgot this)

TCPFLOW 1.4.0
Input: /corp/nps/packets/2008-nitroba/nitroba.pcap
Generated: 2013-10-06 21:20:44

Packets analyzed: 91,144 (55.02 MB)
Transports: IPv4 100%

Labeling is important for repeatability & admissibility
The time-based histogram shows how many packets were received, and when they were received.

The histogram lets an analyst make a rapid determination about what’s in a PCAP file.
Port histograms that show sources & destinations. These use the same code as the time-based histogram.

Note:
- Color key for the time histogram is presented on the port histogram.
Address histogram shows source & destination addresses.

Problem: There might be $2^{32}$ IPv4 addresses or $2^{128}$ IPv6 addresses.

Solution: Tree-based counter

- Note 192.168.8.0/21 in above display — the /21 was automatically determined
- The tree has some performance problems that need to be addressed
PDF typically contains vector graphics, allowing for high resolution.

PDF supports 32 bit floats & ints

8.5” ÷ 2^{32} ≈ 1nm feature size

Other vector graphic formats:
- PostScript (PDF is based on PS)
- Windows Meta File (WMF)
- Scalable Vector Graphics (SVG)
The goal of netviz is to give rapid “situational awareness” about a set of packets.

This flow has:

- No TCP
- 9% IPv6
- A big gap with no data
This flow has a single download of about a megabyte. It also has two failed HTTP requests.
MIT ID’99 IDS evaluation.
The graph shows peculiarities of the traffic.

- Lots of FTP
- Not enough off-peak data.
- Lots of non-TCP
MIT ID’99 IDS evaluation. The graph shows peculiarities of the traffic.

- Lots of FTP
- Not enough off-peak data.
- Lots of non-TCP
2 minutes of packets on a high-volume network

- Mostly HTTP & HTTPS
- Mostly to 192.168.128.2

TCPFLOW 1.4.0
Input: sun.pcap
Generated: 2013-09-05 20:35:09
Date range: 2013-09-03 21:31:40 -- 2013-09-03 21:33:23
Packets analyzed: 62,088 (38.54 MB)
Transports: Other 100%

1 minute, 44 seconds (1 second intervals)

Top Source Addresses
1) 67.69.197.51 - 4.28 MB (11%)
2) 184.168.156.1 - 2.80 MB (7%)
3) 192.168.128.2 - 2.58 MB (6%)

Top Destination Addresses
1) 192.168.128.2 - 35.95 MB (93%)
2) 74.125.131.94 - 493.82 KB (1%)
3) 67.69.197.51 - 172.48 KB (0%)

Top Source Ports
1) 80 - 31.07 MB (80%)
2) 443 - 4.89 MB (12%)
3) 57368 - 108.75 KB (0%)

Top Destination Ports
1) 57535 - 1.55 MB (4%)
2) 57390 - 1.33 MB (3%)
3) 80 - 1.30 MB (3%)

![Network Traffic Diagram]
Glenn Henderson’s MS thesis (NPS Sept 2013) applied this visualization to disk images (BE windirs.txt)

Key improvements:

- Zooms area of interest.
- Improved legends.
Same disk image viewed with Autopsy Timeline (beta)
TCPFLOW 1.4.0

Input: /corp/mitll/packets/ideval99/week2/tuesday/inside.tcpdump

Generated: 2013-09-24 15:32:50

Date range: 1999-03-09 08:00:01 -- 1999-03-10 03:03:20

Packets analyzed: 1,571,748 (373.65 MB)

Transports: IPv4 100%

19 hours, 3 minutes (12 minute intervals)

- FTP
- SSH
- Port 23
- Port 25
- HTTP
- HTTPS
- Port 12417
- Port 24777

Top Source Addresses

0 B

16 MB

- 0%
- 100%

172.16.112.194
- 172.16.112.100
- 206.128.0.0/9
- 209.67.29.11
- 172.16.114.148
- 205.176.0.0/13
- 172.16.114.50
- 208.0.0.0/8
- 136.0.0.0/6
- 207.25.71.141

Top Destination Addresses

0 B

25 MB

- 0%
- 100%

172.16.117.64/26
- 172.16.113.105
- 172.16.113.204
- 172.16.112.207
- 172.16.112.194
- 172.16.116.44
- 172.16.114.207
- 172.16.116.128/25
- 172.16.114.148
- 172.16.113.84

Top Source Ports

0 B

240 MB

- 0%
- 100%

80
- 23
- 20
- 22
- 25
- 24777
- 12417
- 21882
- 17377
- 13291

Top Destination Ports

0 B

30 MB

- 0%
- 100%

80
- 23
- 25
- 22
- 24604
- 7496
- 28262
- 19440
- 15481
- 8933

1) 80 - 239.63 MB (65%)
2) 23 - 26.32 MB (7%)
3) 20 - 23.17 MB (6%)

Histograms vs. CDFs
Problem with histograms: # of bins changes the results.
Histogram with 70 bins
We would prefer that the bin count not change the results.
A cumulative distribution function (CDF) plot is less sensitive to bin count.

A CDF shows the fraction of measurements less than a value. Many people find CDFs hard to understand.

CDFs are easy to make with matplotlib:

```python
plt.hist(speed_vals, bins=bins, weights=speed_secs, cumulative=True, histtype='step', normed=True, color='red', linewidth=4)
```
I like overlaying the CDF on a histogram. This is easy to do with matplotlib.

```python
fig, ax1 = plt.subplots()
ax1.hist(speed_vals, bins, weights=speed_secs, log=False)
ax2 = ax1.twinx()
ax2.set_ylim([0, 1])
ax2.hist(speed_vals, bins, weights=speed_secs, cumulative=True, histtype='step', normed=True, color='red', linewidth=4)
plt.show()
```
The overlay on tcpflow is subtle. We hope people can figure it out (but we haven’t tested).
Network Visualization
GraphViz is an easy tool for network visualization

Originally developed by Bell Labs
Multiple layout engines
Simple “language” for describing graphs

digraph G {
    A -> B;
}

$ dot -Tpdf demo1.dot -o demo1.pdf
GraphViz allows tremendous flexibility

You can change:

- Object shapes, colors
- Layout algorithm

```plaintext
digraph G {
  C [shape=star,label="Forensics",height=2];
  Training [shape=box,style=filled,fillcolor=yellow];
  C -> Training;
  C -> Tools;
  C -> Preparation;
  C -> Convictions [label="Legal Authority"];}
```
Graphviz offers several different layout engines. Layout is hard — especially for forensic data

We have a lot of extraneous information

Consider a hypothetical case:

- drive #1 — 10 distinct email addresses
- drive #2 — 20 distinct email addresses
- drive #3 — 30 distinct email addresses
- hacker1 — common between drive #1, #2
- hacker2 — common between drive #1, #2, #3

```python
drive1_emails = ['user%d@drive1' % i for i in range(10,20)]
drive2_emails = ['user%d@drive2' % i for i in range(10,30)]
drive3_emails = ['user%d@drive3' % i for i in range(10,40)]

for drive in [drive1_emails,drive2_emails]:
    drive += ['hacker1']

for drive in [drive1_emails,drive2_emails,drive3_emails]:
    drive += ['hacker2']
```
$ dot -Tpdf emailgraph.dot -o emailgraph-dot.pdf
(filter for drawing directed graphs; best with hierarchies.)
$ neato -Tpdf emailgraph.dot -o emailgraph-neato.pdf
Filter for drawing undirected graphs using spring models.
$ twopi -Tpdf emailgraph.dot -o emailgraph-twopi.pdf
(Radial layout engine.)
$ circio -Tpdf emailgraph.dot -o emailgraph-twopi.pdf

(Circular layout engine.)
$ fdp -Tpdf emailgraph.dot -o emailgraph-fdp.pdf
(Undirected graphs using “spring model.”)
$ sfdp -Tpdf emailgraph.dot -o emailgraph-sfdp.pdf

(Large undirected graphs with spring model.)
Improve the graph by *removing* information.

$ sfdp -Tpdf emailgraph.dot -o emailgraph-sfdp.pdf

- Remove emails from nodes that do not connect
- sfdp is non-deterministic — four runs, four different graphs:
I visualized the same dataset with 3DJS:

Because 3DJS is interactive, you can “fix” the layout.

The only way to save this is printing to PDF and screen capture.
- Doesn’t work well for very complex datasets
The IP carving “drives” visualization — A small number of important connections.

Nodes:
- Hard drives: [IN10-0047.E01]
- MAC addresses: [00:1E:A6:01:9E:3A]

Edges:
- {drive} -> {MAC} — when {MAC} found on {drive}

Selection:
- {MAC} : only if on more than one {drive}
- {drive} : only if linked to a selected {MAC}
Conclusion: Data-driven visualization are an important growth area for open source forensics.

Specific requirements for forensic visualization:
- Repeatability
- No requirement for manual editing
- Ingests large amount of data
- PDF output

We need a “vocabulary” of forensic visualizations:
- Histograms w/ CDF overlay.
- Bar graphs
  - total vs. distinct.
  - identify outliers with split axis vs. logarithmic

Useful tools:
- matplotlib
- graphiviz
- 3DJS — Requires a browser

The role of browser-based visualization are unclear.
- Lots of good technology, but it may not fit our workflow.
FTK5 (1/4)

AccessData Visualization is another market innovation brought to you by the leader in forensic technology, AccessData. Available with FTK out of the box, Visualization allows you to automatically construct timelines and graphically illustrate relationships among parties of interest in a case. It's yet another powerful way for you to improve your efficiency and accuracy, while enriching your reporting capabilities.
AccessData Visualization is another market innovation brought to you by the leader in forensic technology, AccessData. Available with FTK out of the box, Visualization allows you to automatically construct timelines and graphically illustrate relationships among parties of interest in a case. It's yet another powerful way for you to improve your efficiency and accuracy, while enriching your reporting capabilities.
No longer are investigators, forensic analysts and researchers forced to rely on third-party tools like Analyst Notebook, Microsoft Excel or difficult-to-learn opensource software to visualize relationships in between data elements. FTK® can now provide a vivid and intuitive view into case facts, enabling rapid decision making and reducing time to resolution.

File Visualization

accessed file dates to quickly identify gaps or areas of interest.

and makeup.

interactive interface.

reports and case files.

Categories Distribution Chart
July 12, 2001 - October 06, 2001

Legend
- Documents
- Executable
- Graphics
- Spreadsheets
- Unknown Types
- Other Known Types
- Archives

22%
29%
18%
1%
11%
12%
8%
File Visualization reducing time to resolution.

rapid decision making and
into case facts, enabling
vivid and intuitive view

FTK visualize relationships in between data elements.

Microsoft Excel or difficult-to-learn opensource software to
forced to rely on third-party tools like Analyst Notebook,
No longer are investigators, forensic analysts and researchers

size and location.

the target machine for an understanding of relative file

reports and case files.

interactive interface.

and makeup.

accessed file dates to quickly identify gaps or areas of

interest.

Copyright Access Data 2013
Don’t use pie charts!

Thanks to Stephen Few
Avoid pie charts.

Pie charts are colorful, but are poor for comparing numeric data.

Re-ordering a pie chart can influence perception.

—“Save the Pies for Dessert,” Stephen Few, August 2007
Instead of labeled pie charts, use bar graphs

The bar graph makes direct comparison easy!

Copyright © 2007 Stephen Few, Perceptual Edge

You might be inclined to believe that you can do a better job than most in judging differences in values that are encoded as 2-D area. Here's a simple test. If the area of the small circle below equals a value of 1, what is the area of the large circle?

Not easy, is it? When I ask students to guess the size of the large circle, I get answers ranging from around 6 to 50. The area of the large circle is actually 16 times the area of the small circle.

Stephen Kosslyn writes:

The systematic distortion of area is captured by “Steven’s Power Law,” which states that the psychological impression is a function of the actual physical magnitude raised to an exponent (and multiplied by a scaling constant). To be precise, the perceived area is usually equal to the actual area raised to an exponent of about 0.8, times a scaling constant. In contrast, relative line length is perceived almost perfectly, provided that the lines are oriented the same way. (Kosslyn, Stephen, Graph Design for the Eye and Mind, Oxford University Press, 2006, p. 40)

Perhaps you object to the fact that you had to rely on relative 2-D areas alone to discern the differences above, without the benefit of relative angles as well, which play a role in pie charts. Here’s another test, this time using an actual pie chart. Look at the pie chart below and try to place the slices in order from largest to smallest.

Naomi Robbins writes:

We make angle judgments when we read a pie chart, but we don’t judge angles very well. These judgments are biased; we underestimate acute angles (angles less than 90°) and overestimate obtuse angles (angles greater than 90°). Also, angles with horizontal bisectors (when the line dividing the angle in two is horizontal) appear larger than angles with vertical bisectors. (Naomi Robbins, Creating More Effective Graphs, Wiley, 2005, p. 49)

If a chart is doing its job, you shouldn’t have to struggle. Look at how easy it is to compare the percentages using the bar graph below, which displays the same values:

The bar graph makes direct comparison easy!
Never use 3D effects — they distort relationships. (The distortion changes with different 3D projections.)

People love dressing up their pie charts today to look mouthwatering. Why stick with a simple 2-D pie chart when you can add a third dimension of depth to the picture and throw in some lighting effects and contoured edges while you're at it, as shown here?

It's pretty and eye-catching, but is it more meaningful or easier to interpret? Actually, by adding depth to the pie and changing its angle, we've made it more difficult to interpret. The green slice now appears greater than it actually is, because of the depth that's been added. The slices are now more difficult to compare, because the angle skews their appearance.

For those of you who can't resist tilting your pies (after all, pie tilting is an ancient and respected sport among cultures known for their talent with pastries), let me illustrate the effect that you're creating. Below are three pie charts that are exactly the same, except that the one above is 2-D and the other two are 3-D and tilted. Notice how different the relationships between the slices appear from one version to the next.

Why stop here? Based on what many software vendors advertise, we're encouraged to tweak them out with abandon. For example, with Excel and several other products, you can now easily manipulate the transparency of the pie, creating utterly useless charts such as this:

Believe it or not, this is the same pie chart as the one above. The only difference is that now you can see through it. Aren't you happy Microsoft added this nifty feature? Trust me when I say that I am not doing anything here that isn't marketed with pride by countless software vendors. Here's a Crystal Xcelsius pie chart that's been polished to a high-gloss gleam:

I pulled this example from a dashboard. One of the objectives of a dashboard is to present information in a way that can be quickly read and easily understood. If you glance at this too quickly, however, you're liable to think that it contains three slices. This misperception is a result of the simulated reflection of light on the shiny surface of the pie. When light reflects like this off of objects in the real world, we find it annoying. We have to squint to block the glare in an effort to see the object clearly. Why would we ever want to reproduce this annoying and misleading effect on a computer screen?

Sometimes, despite their obvious limitations, absurd demands are placed on pie charts to show a great deal more than usual. On the following page there is an example from Advizor Analyst/X (an otherwise good product), which attempts to show two levels of part-to-whole relationships at one time: one per country (the slices) and one per product type (the circular bands of color within each country). It would be impossible to compare quantities of a product type between countries, given how differently they are shaped.
There’s a lot of work in visualizations — but few translate to open source software.

In most of the academic world, success is a publication.

To sustain forensic visualizations, they must be built into open source software that is used and maintained.

Fig. 5: BGP-Event-Visualization: The pixel visualization on the left acts as an overview to be able to focus on interesting events (e.g., AS31733 with a high Z-Score). The graph visualization with an underlying geographic map reveals details about the selected route. Grey paths are obsolete, red paths are new routings.

"Visual Analytics for BGP Monitoring and Prefix Hijacking Identification"
Fabian Fischer @ University of Konstanz
— http://www.vis.uni-konstanz.de/en/members/fischer/

To sustain forensic visualizations, they must be built into open source software that is used and maintained.