

FloCon[®]2014



The Rayon Tools: Visualization at the Command Line

A useful visualization must integrate cleanly into an analyst's work environment. For many network security analysts, that environment is the UNIX command line. Rayon provides visualization that works well with the workflow model of UNIX and the shell.

Rayon is a Python library and a set of command-line tools. An analyst can use the Rayon tools in a UNIX command pipeline to visualize data after selecting and transforming it with other UNIX utilities.

What's so Great about the Shell?

The shell is common. UNIX is a pervasive operating system in network analysis environments, the shell is the most basic way of interacting with a UNIX system, and is supported on virtually all of them.

The shell is open. An application needs to do very little to work in the shell environment. The user can compose shell-friendly tools into scripts or command pipelines that are much more useful than any of the tools in isolation.

The shell is automatable. Users write shell scripts with the same commands they use to manually work in the shell. They can write more advanced tools using programming languages (e.g., Python, C), and use those tools, in turn, in other scripts or command pipelines, incrementally automating larger and more complex tasks.

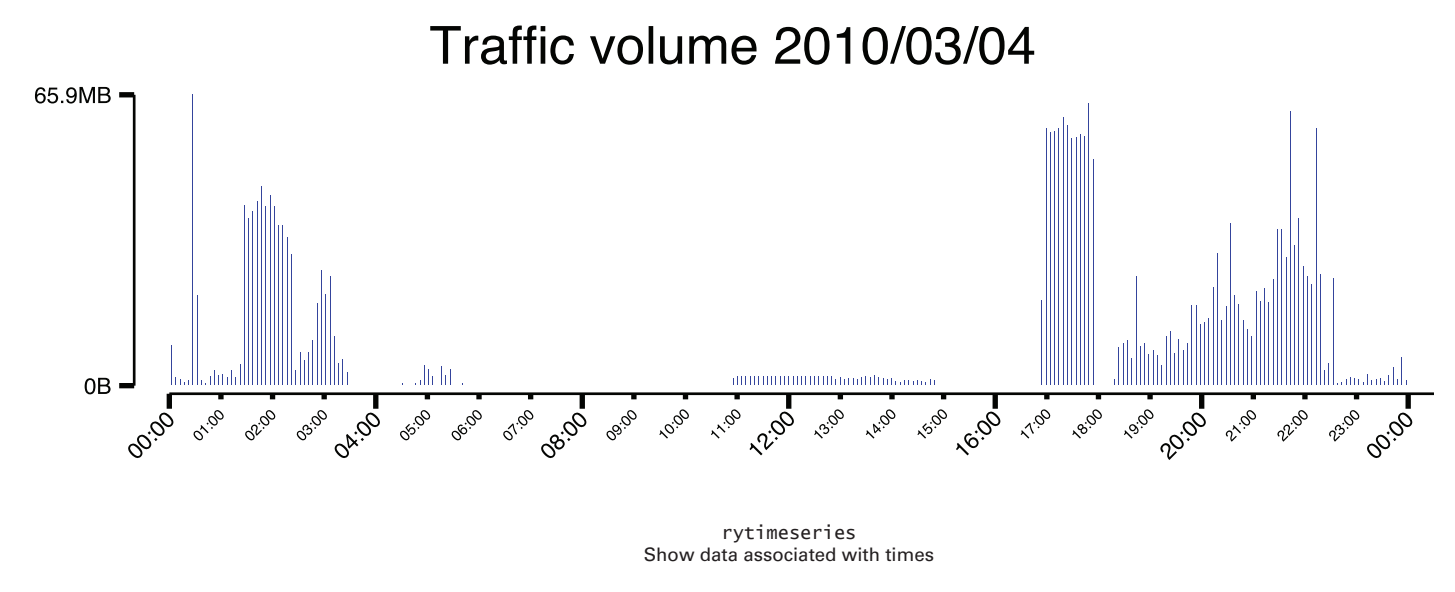
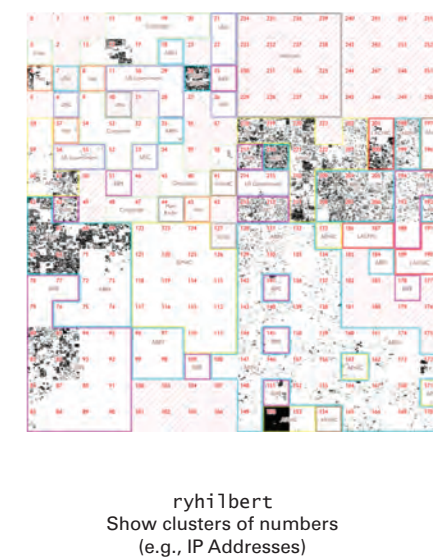
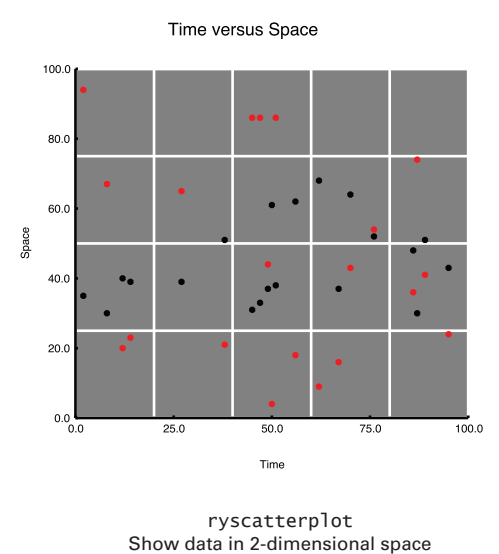
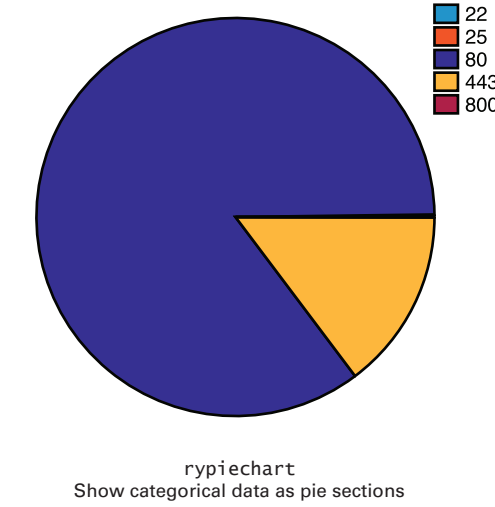
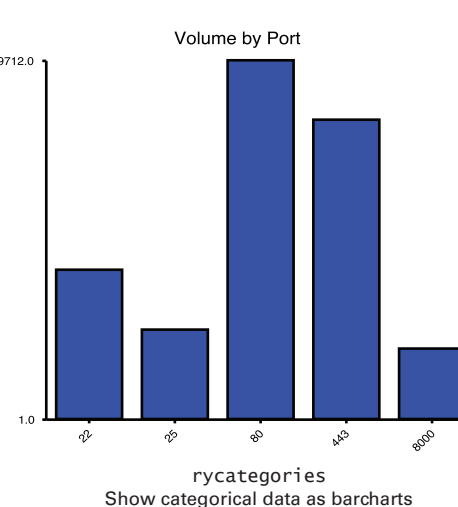
Case Study: Server Traffic

The SiLK tool `rwcount` generates time series from netflow data. The following command extracts all traffic inbound to ports 80 and 443 (nominally, all incoming traffic to web servers) within a time range:

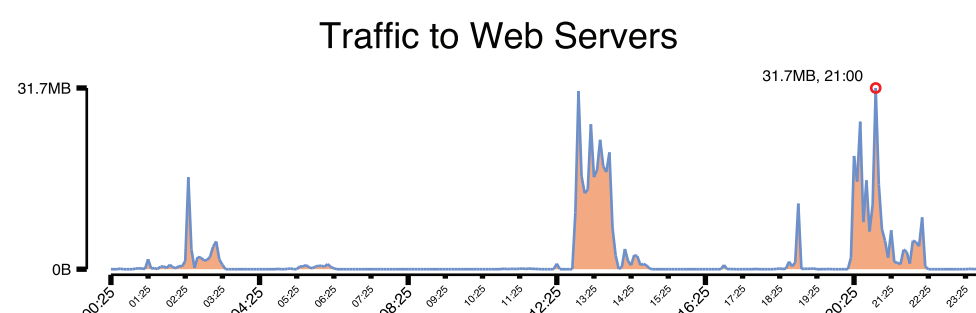
```
rwfilter --start-date=2005/01/07:00 \
--end-date=2005/01/07:23 \
--type=in --proto=6 \
--dport=80 --pass=stdout | \
rwcount --bin-size=300
```

The following examples illustrate incrementally more complex uses of the Rayon tool `rytimeseries`, as it might be used in a simple analysis.

The Rayon tools

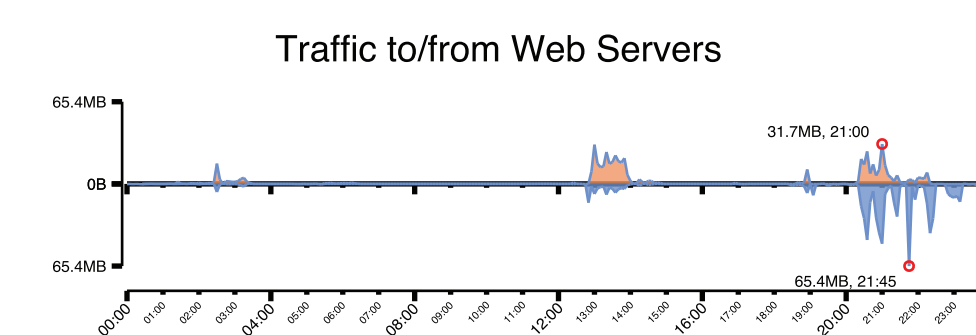


1: How much traffic is coming to web servers?



```
rwfilter \
--start-date=2005/01/07:00 --end-date=2005/01/07:23 \
--proto=6 --type=in,inweb --dport=80 --pass=stdout | \
rwcount --bin-size=300 --no-titles --delimited | \
rytimeseries \
--style=filled_lines --output-path="1.pdf" \
--first-line-colnames --top-column="Bytes" \
--value-tick-label-format=metric --value-units=B \
--annotate-max \
--title="Traffic to Web Servers"
```

2: How does inbound traffic compare with outbound?

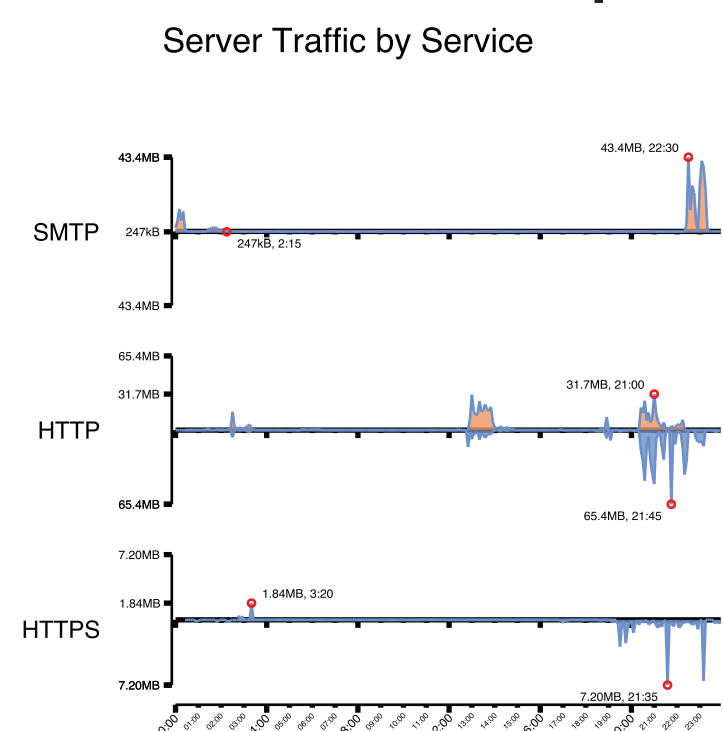


```
rwfilter \
--start-date=2005/01/07:00 --end-date=2005/01/07:23 \
--type=in,inweb --proto=6 --dport=80 --pass=stdout | \
rwcount --bin-size=300 --no-titles --delimited | \
awk -F\| '{printf("%s|s|in\n", $1, $3)}' > 2-top.txt

rwfilter \
--start-date=2005/01/07:00 --end-date=2005/01/07:23 \
--type=out,outweb --proto=6 --sport=80 --pass=stdout | \
rwcount --bin-size=300 --no-titles --delimited | \
awk -F\| '{printf("%s|s|out\n", $1, $3)}' > 2-btm.txt

cat *.txt | rytimeseries \
--style=filled_lines --output-path=2.pdf \
--top-filter="[2]=in" --bottom-filter="[2]=out" \
--top-column=1 --bottom-column=1 \
--annotate-max \
--value-tick-label-format=metric --value-units=B \
--title="Traffic to/from Web Servers"
```

3: How do different services compare?



```
for port in 25 80 443; do
  rwfilter \
  --start-date=2005/01/07:00 --end-date=2005/01/07:23 \
  --proto=6 --type=in,inweb --dport=$port --pass=stdout | \
  rwcount --bin-size=300 --no-titles --delimited | \
  awk -F\| '{printf("%s|s|in|${port}\n", $1, $3)}' \
  > 3-top-{$port}.txt
done

rwfilter \
--start-date=2005/01/07:00 --end-date=2005/01/07:23 \
--proto=6 --type=out,outweb --sport=$port --pass=stdout | \
rwcount --bin-size=300 --no-titles --delimited | \
awk -F\| '{printf("%s|s|out|${port}\n", $1, $3)}' \
> 3-btm-{$port}.txt;

cat *.txt | rytimeseries \
--style=filled_lines --output-path=3.pdf \
--top-filter="[2]=in" --bottom-filter="[2]=out" \
--top-column=1 --bottom-column=1 \
--value-ticks=max,smax \
--value-tick-label-format=metric --value-units=B \
--annotate-max --group-by=3 \
--show=25,80,443 --labels="SMTP,HTTP,HTTPS"
```

Applications

Exploratory analysis. Generating visualizations at the command line keeps all the data in one place for analysis, lets users stay in a single workflow environment longer, and keeps the data in a central location. Users can view the visualizations on their client using X Windows or their web browser.

Prototyping. Users can create visualizations while developing data analysis techniques. These visualizations may eventually become work products, or they can be used to check the results of the analysis for validity.

Automation. Producing reports takes up a lot of analysts' time. Analysts can use Rayon in conjunction with publication tools such as LaTeX or HTML to automatically generate reports, freeing time for other uses.

Visualization/Analysis as a Service. Analysts can share their analyses (including visualizations) on the web. Other users can call them directly as needed, or the analyses can run at scheduled intervals and make their results available online.

Phil Groce, CERT Engineering
<pgroce@cert.org>

Copyright 2013 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Homeland Security and the Defense Information Systems Agency under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use: Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use: This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use.

Requests for permission should be directed to the Software Engineering Institute at: permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.
CERT® is a registered mark of Carnegie Mellon University.
DM-0000781