Network Analysis with SiLK

Ron Bandes

SEI/CERT Network Situational Awareness
### Network Analysis with SiLK

**Author(s):** Bandes / Ron

**Performing Organization:**
Software Engineering Institute Carnegie Mellon University  
Pittsburgh, PA 15213

**DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release, distribution unlimited.

**ABSTRACT**

**SUBJECT TERMS**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
<td>Unclassified</td>
<td>Unclassified</td>
<td>Unclassified</td>
</tr>
</tbody>
</table>

**NUMBER OF PAGES:** 141
This material is based upon work funded and supported by the Department of Defense 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.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

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.

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

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University.

DM-0002038
Outline — 1

Introduction: SiLK
Network flow
Basic SiLK tools
Advanced SiLK tools
Summary
What is Network Flow?

- A log of all network activity
- Not a recording of all packets
- A record of metadata from related packets
  - Similar to a phone bill (call detail record)
- Content of messages is *not* recorded
  - Much, much more compact
    - Can retain longer
    - Less processing
- Increased privacy
What SiLK Does

Retrospective analysis

- most useful for analysing past network events
- may feed an automated report generator
- good for forensics (what happened before the incident?)

Descriptive analysis – profiling/categorizing

Exploratory analysis – looking for the unusual

Optimized for extremely large data collections

- Very compact record format
- Large amount of history can stay online.
- Can be processed much more quickly than packets
# Modes of Inquiry

<table>
<thead>
<tr>
<th>Detect</th>
<th>Discover</th>
</tr>
</thead>
<tbody>
<tr>
<td>example: Snort®</td>
<td>example: SiLK™</td>
</tr>
<tr>
<td>Operate like Tech Support</td>
<td>Operate like Quality Assurance</td>
</tr>
<tr>
<td>Response</td>
<td>Exploring</td>
</tr>
<tr>
<td>Alert</td>
<td>Question</td>
</tr>
<tr>
<td>Relax between detections</td>
<td>Continuous security improvement</td>
</tr>
<tr>
<td></td>
<td>Produce new indicators</td>
</tr>
<tr>
<td></td>
<td>Shorten detect/response time</td>
</tr>
</tbody>
</table>

Snort is a registered trademark of Cisco and/or its affiliates
SiLK is a trademark of Carnegie Mellon University
Got a Question? Flow Can Help

What’s on my network?

What happened before the event?

Where are policy violations occurring?

What are the most popular web servers?

By how much would volume be reduced with a blacklist?

Do my users browse to known infected web servers?

Do I have a spammer on my network?

When did my web server stop responding to queries?

Who uses my public servers?
Outline — 2

Introduction: SiLK

Network flow

Basic SiLK tools

Advanced SiLK tools

Summary
Unidirectional Flows (Uniflows)
Packet Encapsulation

- Ethernet frame
  - Dest MAC address
  - Source MAC address
  - Type of packet

- IP datagram (packet)
  - Src IP address
  - Dst IP address
  - Type of segment

- Transport segment
  - Src port
  - Dest port
  - Flags

- Application layer message (HTTP, SMTP, DNS)
Two TCP/IP Sockets Make a Connection

TCP/IP SOCKET
IP address: 10.0.0.1
L4 protocol: TCP
High-numbered ephemeral port #

TCP/IP SOCKET
IP address: 203.0.113.1
L4 protocol: TCP
Low-numbered Well-Known-Port #

Client  Connection  Server
Network Flow versus NetFlow

Network Flow—a generic term for the summarization of packets related to the same flow or connection into a single record

NetFlow™—A Cisco trademarked set of format specifications for storing network flow information in a digital record

IPFIX—a format specification from the IETF for flow records, an extension of Cisco NetFlow v9

SiLK—Another set of format specifications for flow records and other related data, plus the tool suite to process that data
What’s in a Record?

Fields found to be useful in analysis:

- source address, destination address
- source port, destination port (Internet Control Message Protocol [ICMP] type/code)
- IP [transport] protocol
- bytes, packets in flow
- accumulated TCP flags (all packets, first packet)
- start time, duration (milliseconds)
- end time (derived)
- sensor identity
- flow termination conditions
- application-layer protocol
DNS packets viewed in Wireshark

Wireshark is a registered trademark of the Wireshark Foundation
Sequence Diagram

DNS Client
192.168.1.105
UDP port 50744

DNS Server
10.1.10.1
UDP port 53

Request (type A)

Response (type A)
## SiLK tool (rwcut) output

<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP</th>
<th>sPort</th>
<th>dPort</th>
<th>pro</th>
<th>packets</th>
<th>bytes</th>
<th>sensor</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.105</td>
<td>10.1.10.1</td>
<td>50744</td>
<td>53</td>
<td>17</td>
<td>1</td>
<td>64</td>
<td>S1</td>
<td>out</td>
</tr>
<tr>
<td>10.1.10.1</td>
<td>192.168.1.105</td>
<td>53</td>
<td>50744</td>
<td>17</td>
<td>1</td>
<td>80</td>
<td>S1</td>
<td>in</td>
</tr>
</tbody>
</table>
Network Monitoring

Internet

Other internetwork

sensor

sensor

sensor

SiLK repository

SiLK

Console

terminal

iSiLK

iSiLK is a trademark of Carnegie Mellon University
Realistic Sequence Diagram

DNS Client
192.168.1.105
UDP port 50744

Local Server
10.1.10.1

Sensor

Root Server
.com Server
.mudynamics.com Server

Request (type A)
Dest port 53

Response (type A)
Src port 53
More Realistic Sequence Diagram

DNS Client
192.168.1.105
UDP port 50744

Local Server
10.1.10.1

NAT

Sensor

Root Server

Server

Server

.com

Root

Server

Sensor

NAT

Request (type A)
Dest port 53

Response (type A)
Src port 53

.com Server

.mudynamics.com Server
<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP</th>
<th>sPort</th>
<th>dPort</th>
<th>pro</th>
<th>packets</th>
<th>flags</th>
<th>initF</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.105</td>
<td>10.1.10.1</td>
<td>50744</td>
<td>53</td>
<td>17</td>
<td>1</td>
<td></td>
<td></td>
<td>out</td>
</tr>
<tr>
<td>10.1.10.1</td>
<td>192.168.1.105</td>
<td>53</td>
<td>50744</td>
<td>17</td>
<td>1</td>
<td></td>
<td></td>
<td>in</td>
</tr>
<tr>
<td>192.168.1.105</td>
<td>198.51.100.6</td>
<td>49152</td>
<td>80</td>
<td>6</td>
<td>4</td>
<td>SRPA</td>
<td>S</td>
<td>outweb</td>
</tr>
<tr>
<td>198.51.100.6</td>
<td>192.168.1.105</td>
<td>80</td>
<td>49152</td>
<td>6</td>
<td>3</td>
<td>S PA</td>
<td>S A</td>
<td>inweb</td>
</tr>
</tbody>
</table>
HTTP Sequence Diagram

HTTP Client 192.168.1.105
HTTP Server 198.51.100.6
DNS Server 10.1.10.1

DNS Request (type A)
DNS Response (type A)

SYN
SYN, ACK
ACK
GET
ACK
Okay
RST
## What Is This? — 2

<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP</th>
<th>sPort</th>
<th>dPort</th>
<th>pro</th>
<th>packets</th>
<th>bytes</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.22.105.250</td>
<td>71.55.40.253</td>
<td>52415</td>
<td>25</td>
<td>6</td>
<td>22</td>
<td>14045</td>
<td>FSRPA</td>
</tr>
<tr>
<td>71.55.40.253</td>
<td>30.22.105.250</td>
<td>25</td>
<td>52415</td>
<td>6</td>
<td>19</td>
<td>1283</td>
<td>FS PA</td>
</tr>
<tr>
<td>30.22.105.250</td>
<td>71.55.40.253</td>
<td>52415</td>
<td>25</td>
<td>6</td>
<td>1</td>
<td>40</td>
<td>R</td>
</tr>
<tr>
<td>sIP</td>
<td>dIP</td>
<td>pro</td>
<td>packets</td>
<td>bytes</td>
<td>sTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-----</td>
<td>---------</td>
<td>-------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.89</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:04:30.172</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.149.249</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:04:37.302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.52</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:04:37.312</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.127</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:04:58.363</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.196</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:04.327</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.149.30</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:09.242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.149.173</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:12.174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.13</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:14.114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.56</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:15.383</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.114</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:18.228</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.202.92</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:22.466</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.202.68</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:23.497</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.24.161</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:30.256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.217.139.155</td>
<td>177.252.202.238</td>
<td>1</td>
<td></td>
<td>122</td>
<td>2010/12/08T00:05:33.088</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### What Is This? — 4

<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP</th>
<th>sPort</th>
<th>dPort</th>
<th>pkts</th>
<th>bytes</th>
<th>flags</th>
<th>sTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.187.13.78</td>
<td>71.55.40.204</td>
<td>40936</td>
<td>80</td>
<td>83</td>
<td>3512</td>
<td>FS PA</td>
<td>2010/12/08T11:00:01</td>
</tr>
<tr>
<td>71.55.40.204</td>
<td>88.187.13.78</td>
<td>80</td>
<td>40936</td>
<td>84</td>
<td>104630</td>
<td>FS PA</td>
<td>2010/12/08T11:00:01</td>
</tr>
<tr>
<td>88.187.13.78</td>
<td>71.55.40.204</td>
<td>40938</td>
<td>80</td>
<td>120</td>
<td>4973</td>
<td>FS PA</td>
<td>2010/12/08T11:00:04</td>
</tr>
<tr>
<td>71.55.40.204</td>
<td>88.187.13.78</td>
<td>80</td>
<td>40938</td>
<td>123</td>
<td>155795</td>
<td>FS PA</td>
<td>2010/12/08T11:00:05</td>
</tr>
<tr>
<td>88.187.13.78</td>
<td>71.55.40.204</td>
<td>56172</td>
<td>80</td>
<td>84</td>
<td>3553</td>
<td>FS PA</td>
<td>2010/12/08T12:00:02</td>
</tr>
<tr>
<td>71.55.40.204</td>
<td>88.187.13.78</td>
<td>80</td>
<td>56172</td>
<td>83</td>
<td>103309</td>
<td>FS PA</td>
<td>2010/12/08T12:00:02</td>
</tr>
<tr>
<td>88.187.13.78</td>
<td>71.55.40.204</td>
<td>56177</td>
<td>80</td>
<td>123</td>
<td>5093</td>
<td>FS PA</td>
<td>2010/12/08T12:00:05</td>
</tr>
<tr>
<td>71.55.40.204</td>
<td>88.187.13.78</td>
<td>80</td>
<td>56177</td>
<td>124</td>
<td>157116</td>
<td>FS PA</td>
<td>2010/12/08T12:00:05</td>
</tr>
</tbody>
</table>
It’s All a Matter of Timing

The flow buffer needs to be kept manageable.

Idle timeout

- If there is no activity within 30 seconds (configurable), flush the flow.

Active timeout

- Flush all flows open for 30 minutes (configurable).

```
Flow 1
```

```
Flow 2
```

```
Flow 3
```
SiLK Types

- **Internal network**
  - int2int
  - inweb, inicmp, in
  - outnull

- **External network**
  - outweb, outicmp, out
  - ext2ext
  - innull

- **Null**
  - other*

*to/from network that is neither internal nor external
## SiLK Types in SiLK

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inweb, outweb</td>
<td>Inbound/outbound TCP ports 80, 443, 8080</td>
</tr>
<tr>
<td>innull, outnull</td>
<td>Inbound/outbound filtered traffic</td>
</tr>
<tr>
<td>inicmp, outicmp</td>
<td>Inbound/outbound IP protocol 1</td>
</tr>
<tr>
<td>in, out</td>
<td>Inbound/outbound not in above categories</td>
</tr>
<tr>
<td>int2int, ext2ext</td>
<td>Internal to internal, external to external</td>
</tr>
<tr>
<td>other</td>
<td>Source not internal or external, or destination not internal, external, or null</td>
</tr>
</tbody>
</table>

Names in **bold** are default types
Outline — 3

Introduction: SiLK
Network flow
Basic SiLK tools
Advanced SiLK tools
Summary
UNIX / Linux commands

System prompt
  Info + prompt character
  e.g., ~ 101>

User command
  command name rwfilter (case sensitive)
  options -h --help -k2 --key=2
  arguments results.rw
  redirections > >> <
  pipe |

For example:

  rwcut --all-fields results.rw > results.txt
  rwcut --fields=1-6 results.rw | more
Some standard Linux commands

`ls` – list name & attributes of files and directories
`cd` – change the current working directory
`cat` – output the contents of a file
`more` and `less` – display a file one page at a time
`cut` – output only selected fields of a file
`sort` – reorder the records (lines) of a file
`wc` – word count (optionally, line count) of a file
`exit` – logout & terminate a terminal window
Linux Standard symbolic files

Standard In (\texttt{stdin}) – where normal (especially interactive) input comes from

Standard Out (\texttt{stdout}) – where normal/expected (especially interactive) output goes to

Standard Error (\texttt{stderr}) – where messages (especially unexpected) go to

Defaults:

\begin{itemize}
  \item \texttt{stdin} – keyboard
  \item \texttt{stdout} – screen/window
  \item \texttt{stderr} – screen/window
\end{itemize}

Defaults are overridden by redirections and pipes
Shell Scripts

Put a complicated command, pipeline, or sequence of pipelines into a script file.

- It saves your commands for reuse or learning
- It eases making changes

Use the GUI editor **gedit**, or the simple character editors **joe** and **nano** when on a SSH connection. Use **vi** (**vim**) to earn your geek badge. Vi or vim can be found on every Linux/UNIX system.

Name your shell script something like **dothis.sh**

Execute (run) your script: **./dothis.sh**

gedit is the registered trademark of Interactive Graphic Systems, Inc.
SSH is the registered trademark of SSH Communications Security Corp
Collection, Packing, and Analysis

Collection of flow data

- Examines packets and summarizes into standard flow records
- Timeout and payload-size values are established during collection

Packing stores flow records in a scheme optimized for space and ease of analysis

Analysis of flow data

- Investigation of flow records using SiLK tools
Collection

Idle-timeout, Active-timeout

Termination-attribute, Application, Start-time, Duration, Packets, Bytes, Flags…

 tcpdump

 YAF

 IPFIX

PCAP
Packing

- IPFIX
- Cisco NetFlow

rwflowpack

Sensor, Class, Type

Packing logic plug-in

sensor.conf

SiLK repository
SiLK Repository

- RootDir
  - Sensor0
  - Sensor1
  - silk.conf

- in
- inweb
- int2int
- out
- outweb
- ext2ext

- type-SENSOR_yyyymmdd.hh
  e.g., in-SEN1_20091231.23
Linux Exercise

PS1='\W \!> '  
export SILK_IPV6_POLICY=asv4  
cd /data/bluered  
ls -l silk.conf  
less silk.conf  # type “q” to exit from less  

cd
Analysis

SiLK repository

SiLK tool chain

Raw (binary) flow records in a file

Text

Raw (binary) flow records in a file
Reporting

Text

UNIX text tools (sed, awk, …)

Text

Visualization tools (gnuplot, Rayon, Excel)

Rayon is a trademark of Carnegie Mellon University
Excel is a registered trademark of Microsoft Corporation
So Much to Do, So Little Time...

We can’t discuss all parameters for every tool.

Resources

- Analyst’s Handbook
- SiLK Reference Guide (hard-copy man pages)
- `--help` option
- `man` command
- `<http://tools.netsa.cert.org>`
What sensors are defined?

```bash
rwsiteinfo --fields=id-sensor,sensor
rwsiteinfo --fields=id-sensor,sensor,\
    describe-sensor
```
Basic SiLK Tools: rwfileinfo

rwfileinfo displays a variety of characteristics for each file format produced by the SiLK tool suite.

It is very helpful in tracing how a file was created and where it was generated.
rwfileinfo Example

[liveuser@livecd ~]$ rwfilter --sensor=S0 --type=in,out \   --start=2009/4/21T15 --protocol=1 \   --pass=icmprecords.rw

[liveuser@livecd ~]$ rwfileinfo icmprecords.rw

icmprecords.rw:
  format(id)          FT_RWIPV6ROUTING(0x0C)
  version             16
  byte-order          littleEndian
  compression(id)     lzo1x(2)
  header-length       176
  record-length       88
  record-version      1
  silk-version        3.9.0
  count-records       39
  file-size           963
  command-lines
    1  rwfilter --sensor=S0 --type=in,out \   --start=2009/4/21T15 --protocol=1 --pass=icmprecords.rw
rwfileinfo --fields

All fields available to display

1 format(id)  10 record-version
2 version     11 silk-version
3 byte-order  12 packed-file-info
4 compression(id)  13 probe-name
5 header-length  14 annotations
6 record-length  15 prefix-map
7 count-records  16 ipset
8 file-size     17 bag
9 command-lines
But I can’t read binary...

**rwcut** provides a way to display binary records as human-readable ASCII:

- useful for printing flows to the screen
- useful for input to text-processing tools
- Usually you’ll only need the **--fields** option.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip</td>
<td>Packets</td>
</tr>
<tr>
<td>dip</td>
<td>Type</td>
</tr>
<tr>
<td>sport</td>
<td>Flags</td>
</tr>
<tr>
<td>dip</td>
<td>Bytes</td>
</tr>
<tr>
<td>sport</td>
<td>In</td>
</tr>
<tr>
<td>dport</td>
<td>Initial flags</td>
</tr>
<tr>
<td>protocol</td>
<td>Sensor</td>
</tr>
<tr>
<td>dip</td>
<td>Out</td>
</tr>
<tr>
<td>dport</td>
<td>Session flags</td>
</tr>
<tr>
<td>protocol</td>
<td>DCC</td>
</tr>
<tr>
<td>dip</td>
<td>Dur</td>
</tr>
<tr>
<td>dport</td>
<td>Application</td>
</tr>
<tr>
<td>protocol</td>
<td>Attributes</td>
</tr>
<tr>
<td>sip</td>
<td>Time</td>
</tr>
<tr>
<td>dip</td>
<td>Nhip</td>
</tr>
<tr>
<td>dport</td>
<td>Etim</td>
</tr>
<tr>
<td>protocol</td>
<td>Itype &amp; icode</td>
</tr>
</tbody>
</table>

Field names in italics are *derived* fields.
rwcut Default Display

By default

- sIP, sPort
- dIP, dPort
- protocol
- packets, bytes
- flags
- sTime, eTime, duration
- sensor

--all-fields
## Pretty Printing SiLK Output

Default output is fixed-width, pipe-delimited data.

```
<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP</th>
<th>pro</th>
<th>pkts</th>
<th>bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>207.240.215.71</td>
<td>128.3.48.203</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>207.240.215.71</td>
<td>128.3.48.68</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>207.240.215.71</td>
<td>128.3.48.71</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
</tbody>
</table>
```

Tools with text output have these formatting options:

- **--no-titles**: suppress the column headings
- **--no-columns**: suppress the spaces
- **--column-separator**: just change the bar to something else
- **--delimited**: combine above 3 options
- **--legacy-timestamps**: better for import to Excel
What do the data look like?

rwcut icmprecords.rw --fields=1-6

Try other values for --fields.
Try omitting the --fields option.
Why do we need `rwcut`?

cd

rwfilter --type=in
    --start-d=2009/4/21T15 --proto=0-
    --compress=none
    --pass-dest=t20.rw --max-pass=20

ls -l t20.rw

rwfileinfo t20.rw

hexdump -C t20.rw  # any readable text?

rwcut --fields=1-6 t20.rw
Basic SiLK Tools: rwsort

Why sort flow records?

- Records are recorded as received, not necessarily in time order.
- Analysis often requires finding outliers.
- You can also sort on other fields such as IP address or port to easily find scanning patterns.
- It allows analysts to find behavior such as beaconing or the start of traffic flooding.
rwsort Options

--fields (same as rwcut) is required.

Input files are specified as positional arguments (default is stdin).

--output-path= specifies the output file (default is stdout.)

For improved sorts, specify a buffer size with --sort-buffer-size=.

For large sorts, specify a temporary directory with --temp-directory=.
Temporary files stored in /tmp by default

```
rwsort t20.rw --fields=stime \  --output-path=t20bystime.rw
rwsort t20.rw --fields=sip,sport,dport \ | rwuniq --fields=sip,sport,dport --presorted \  --value=dip-distinct
```
Basic SiLK Tools: **rwfilter**

- Pick files from the repository
- Compression
- Basic statistics
- Advanced flow-by-flow filtering
- Plug in additional tools
- Direct flow output

Swiss Army knife logo is a registered trademark of Victorinox AG
rwfilter Syntax

General form

```bash
rwfilter {INPUT | SELECTION}
PARTITION OUTPUT [OTHER]
```

Example call

```bash
rwfilter --sensor=S0 --type=in \n  --start-date=2009/4/21T9 \n  --end-date=2009/4/21T16 \n  --protocol=0-255 --pass=workday-21.rw
```
The rwfilter command requires three basic parts:

• **selection** criteria or **input** criteria (which files are input?)
  – repository: class, sensor, type, start/end date/hour

• **Partition** (which records pass my criteria? Which fail?)
  – filter options: Which flows do I really want?

• **output** options

Partitioning is the most complex part.
Selection and Input Criteria

Selection options control access to repository files:

- `--start-date=2009/4/21`
- `--end-date=2009/4/21T03`
- `--sensor=S0`
- `--class=all`
- `--type=in,inweb`

Alternatively, use input criteria for a pipe or a file:

- `myfile.rw`
- `stdin`
- useful for chaining filters through stdin/stdout
### --start-date and --end-date

<table>
<thead>
<tr>
<th>--end-date</th>
<th>--start-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour</td>
<td>Day</td>
</tr>
<tr>
<td>Hours in explicit range</td>
<td>Ignore end-date hour. Whole days.</td>
</tr>
</tbody>
</table>
| End-hour is the same as start-hour. 
#hours = 1, 25, 49, … | Whole days. | Error |
| None | 1 hour | 1 day | Current day to present time. |
How Many Files are Selected?

#Files = Sensors 
  x Types 
  x Hours 
  – missing files
rwfilter Partitioning Parameters

Flow Record Fields
IP Sets
User pmaps and Country Codes
Tuples
Plugins
PySiLK
Basic Partitioning Options

• Simple numeric fields: ports, protocol, ICMP Type
• Specified IP addresses, CIDR blocks, & wildcards
• Sets of IP addresses
• Combinations of key fields – Tuples
Simple Numeric Key Fields

--protocol=
--sport= --dport= --aport=  # source, dest, any

--protocol=6,17  # TCP or UDP
--protocol=1-5,7-16,18-  # not TCP or UDP
--protocol=0-  # all protocols
--dport=80,443  # HTTP or HTTPS
--sport=6000-6063,9100-9107  # X11 or JetDirect
--aport=20,21  # FTP
--sport=0-1023  # Well Known Ports
ICMP Types and Codes

--icmp-type major type of ICMP message
--icmp-code sub-type of ICMP message

--icmp-type=0,8 # ping request & reply
--icmp-type=3 --icmp-code=4 # fragm’n needed
Specified IP address, CIDR block, or wildcard

--saddress=  --daddress=  --any-address=
--not-saddress=  --not-daddress=  --not-any-address=

May specify a single:

IP address      192.0.2.1
CIDR block       192.0.2.0/24
wildcard pattern 172.16-31.x.1,254
adrrs in same subnet    203.0.113.1,3,7,13,19
Specified IP addresses or CIDR blocks

--scidr= --dcidr= --any-cidr=
--not-scidr= --not-dcidr= --not-any-cidr=

May specify multiple:

IP addresses 192.0.2.1, 198.51.100.3
CIDR blocks 192.0.2.0/24, 198.51.100.0/24
mixture 192.0.2.1, 192.0.2.8/29
NO wildcard patterns
Sets of arbitrary addresses

--sipset=  --dipset=  --anyset=
--not-sipset=  --not-dipset=  --not-anyset=

Specifies the name of a file storing the IP set:

--sipset=internalservers.set
--dipset=RussianBizNtwk.set
--anyset=TorNodes.set
--not-dipset=whitelist.set
### Combinations of key fields – Tuples

--tuple-file=TorAuthSockets.tuple --tuple-dir=reverse

**TorAuthSockets.tuple file:**

<table>
<thead>
<tr>
<th>sIP</th>
<th>sPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>208.83.223.34</td>
<td>443</td>
</tr>
<tr>
<td>82.94.251.203</td>
<td>80</td>
</tr>
<tr>
<td>193.23.244.244</td>
<td>80</td>
</tr>
<tr>
<td>194.109.206.212</td>
<td>80</td>
</tr>
<tr>
<td>86.59.21.38</td>
<td>80</td>
</tr>
<tr>
<td>128.31.0.34</td>
<td>9131</td>
</tr>
<tr>
<td>171.25.193.9</td>
<td>443</td>
</tr>
<tr>
<td>154.35.32.5</td>
<td>80</td>
</tr>
<tr>
<td>212.112.245.170</td>
<td>80</td>
</tr>
<tr>
<td>76.73.17.194</td>
<td>9030</td>
</tr>
</tbody>
</table>
rwfilter output options

--pass-destination=  # file to get records that pass
--fail-destination=  # file to get records that fail
--all-destination=  # file to get all records

--print-statistics  # report recs read/pass/fail
--print-volume-statistics  # report how many
  # recs/pkts/bytes pass/fail
rwfilter --sensor=S0 --type=in
--start=2009/4/21T00 --end=2009/4/21T07
--daddress=10.1.0.0/16 --print-volume-stat

<table>
<thead>
<tr>
<th></th>
<th>Recs</th>
<th>Packets</th>
<th>Bytes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1436</td>
<td>2615</td>
<td>158084</td>
<td>8</td>
</tr>
<tr>
<td>Pass</td>
<td>1436</td>
<td>2615</td>
<td>158084</td>
<td></td>
</tr>
<tr>
<td>Fail</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
rwfilter exercise

1) Find all traffic captured by sensor S0 going outbound to external HTTPS servers on April 21, 2009. Save these flows in file https0421.rw.

2) How many flow records matched the criteria?
rwfilter exercise solution

```
rwfilter --sensor=S0 --type=outweb \ 
  --start=2009/4/21 --dport=443 \ 
  --pass=https0421.rw --print-volume-statistics

<table>
<thead>
<tr>
<th></th>
<th>Recs</th>
<th>Packets</th>
<th>Bytes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>43656</td>
<td>173550</td>
<td>36174384</td>
<td>24</td>
</tr>
<tr>
<td>Pass</td>
<td>123</td>
<td>1420</td>
<td>288083</td>
<td></td>
</tr>
<tr>
<td>Fail</td>
<td>43533</td>
<td>172130</td>
<td>35886301</td>
<td></td>
</tr>
</tbody>
</table>
```

```
rwfileinfo https0421.rw --fields=count

https0421.rw: count-records 123
```
Output Criteria

rwfilter leaves the flows in binary (compact) form.

- **--pass, --fail**: direct the flows to a file or a pipe
- **--all**: destination for everything pulled from the repository
- One output is required but more than one can be used (no screen allowed).

Other useful output

- **--print-filenames, --print-missing-files**
- **--print-statistics** or **--print-volume-statistics**
Chaining Filters

It is often very efficient to chain `rwfilter` commands together:

- Use `--pass` and `--fail` to segregate bins.
- Use `--all`, so you only pull from the repository once.
rwfilter \  
   --start-date=2010/12/08 \  
   --type=outweb \  
   --bytes=100000- \  
   --pass=stdout \  
| rwfilter \  
   stdin \  
   --duration=60- \  
   --pass=long-http.rw \  
   --fail=short-http.rw
Tips with rwfilter

Narrow time, type, and sensor as much as possible (fewer records to check).

Include as many partitioning parameters as possible (easy to be vague and get too much data).

Can do multiple queries and merge results

Can do further filtering to narrow results

Iterative exploration
## Example Typos

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--port=</td>
<td>No such keywords</td>
</tr>
<tr>
<td>--destport=</td>
<td>No such keywords</td>
</tr>
<tr>
<td>--sip= or --dip=</td>
<td>No such keywords</td>
</tr>
<tr>
<td>--saddress=danset.set</td>
<td>Needs value not filename</td>
</tr>
<tr>
<td>--start-date=2006/06/12--end-date</td>
<td>Spaces needed</td>
</tr>
<tr>
<td>--start-date = 2006/06/12</td>
<td>No spaces around equals</td>
</tr>
<tr>
<td>start-date=2006/06/12</td>
<td>Need dashes</td>
</tr>
<tr>
<td>---start-date=2006/06/12</td>
<td>Only two dashes</td>
</tr>
<tr>
<td>--start-date=2005/11/04:06:00:00  --end-date=2005/05/21:17:59:59</td>
<td>Only down to hour</td>
</tr>
</tbody>
</table>
SiLK Commandments

1. Thou shalt use Sets instead of using several rwfilter commands to pull data for multiple IP addresses.
2. Thou shalt store intermediate data on local disks, not network disks.
3. Thou shalt make initial pulls from the repository, store the results in a file, and work on the file from then on. The repository is slower than processing a single file.
4. Thou shalt work in binary for as long as possible. ASCII representations are much larger and slower than the binary representations of SiLK data.
5. Thou shalt filter no more than a week of traffic at a time. The filter runs for excessive length of time otherwise.
6. Thou shalt only run a few rwfilter commands at once.
7. Thou shalt specify the type of traffic to filter. Defaults work in mysterious ways.
8. Thou shalt appropriately label all output.
9. Thou shalt check that SiLK does not provide a feature before building your own.
Basic SiLK Counting Tools: *rwcount*, *rwstats*, *rwuniq*

“Count [volume] by [key field] and print [summary]”

- basic bandwidth study:
  - “Count bytes by hour and print the results.”

- top 10 talkers list:
  - “Count bytes by source IP and print the 10 highest IPs.”

- user profile:
  - “Count records by dIP-dPort pair and print all the pairs.”

- potential scanners:
  - “Count unique dIPs by sIP and print the sources that contacted more than 100 destinations.”
Bins

For motor vehicle trips we could bin by:

- Vehicle style – sedan, coupe, SUV, pickup, van
- Highway or city trip
- Personal or business trip

We could measure the trips and aggregate in bins:

- Total miles
- Fuel consumption
- Oil consumption
- Pollutant emission

Bins

For flows we could bin by:

- address or address block
- port
- protocol
- time period

We could measure the flows and aggregate in bins:

- count of flow records, packets, bytes
- count of distinct values of other fields, e.g., addr
- earliest sTime, latest eTime
Bins

Packet count

Value
from flow record
e.g., packets

Bin key field
e.g., protocol

Aggregate
Value

Total packets
TCP
Total packets
UDP
Total packets
ICMP

Packet count

Packet count

Packet count

Packet count
Basic SiLK Counting Tools: \texttt{rwcount}, \texttt{rwstats}, \texttt{rwuniq}

\texttt{rwcount}: count volume across time periods

\texttt{rwstats}: count volume across IP, port, or protocol and create descriptive statistics

\texttt{rwuniq}: count volume across any combination of SiLK fields

“Key field” = SiLK fields defining bins

“Volume” = \{Records, Bytes, Packets\} and a few others

measure

aggregate value

Each tool reads raw binary flow records as input.
rwcount

- count records, bytes, and packets by time and display results
- fast, easy way of summarizing volumes as a time series
- great for simple bandwidth studies
- easy to take output and make a graph with graphing S/W

http://www.cs.uoregon.edu/research/tau/docs/paraprof/ch05s02.html
Time Bins

When binning by time, you must specify the period of time for each bin. This is called the **bin-size**. It’s the size of the bin’s opening, not the volume of the container.
rwcount

The bin key is always time. You choose the period. The aggregate measures are chosen for you. They are flows/records, bytes, packets.

```
rwfilter --sensor=S0 --start=2009/4/21 \ 
   --type=in --proto=1 --pass=stdout \ 
| rwcount --bin-size=3600
```

<table>
<thead>
<tr>
<th>Date</th>
<th>Records</th>
<th>Bytes</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009/04/21T13:00:00</td>
<td>10.00</td>
<td>2460.00</td>
<td>41.00</td>
</tr>
<tr>
<td>2009/04/21T14:00:00</td>
<td>29.00</td>
<td>8036.00</td>
<td>107.00</td>
</tr>
<tr>
<td>2009/04/21T15:00:00</td>
<td>22.00</td>
<td>2214.00</td>
<td>47.00</td>
</tr>
<tr>
<td>2009/04/21T16:00:00</td>
<td>10.00</td>
<td>1586.00</td>
<td>23.00</td>
</tr>
</tbody>
</table>

...
**What Is This? — 9**

rwcount MSSP.rw --bin-size=3600

<table>
<thead>
<tr>
<th>Date</th>
<th>Records</th>
<th>Bytes</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010/12/08T00:00:00</td>
<td>1351571.66</td>
<td>73807086.40</td>
<td>1606313.61</td>
</tr>
<tr>
<td>2010/12/08T01:00:00</td>
<td>1002012.43</td>
<td>54451440.59</td>
<td>1185143.62</td>
</tr>
<tr>
<td>2010/12/08T02:00:00</td>
<td>1402404.61</td>
<td>77691865.26</td>
<td>1675282.27</td>
</tr>
<tr>
<td>2010/12/08T03:00:00</td>
<td>1259973.65</td>
<td>68575249.90</td>
<td>1491393.08</td>
</tr>
<tr>
<td>2010/12/08T04:00:00</td>
<td>939313.56</td>
<td>51410968.24</td>
<td>1118584.81</td>
</tr>
<tr>
<td>2010/12/08T05:00:00</td>
<td>459564.75</td>
<td>80862273.32</td>
<td>1742058.62</td>
</tr>
<tr>
<td>2010/12/08T06:00:00</td>
<td>1280651.23</td>
<td>69881126.41</td>
<td>1519435.24</td>
</tr>
</tbody>
</table>

...
rwcount Demo

The shell can help with the arithmetic: $(24*60*60)$
You also can find common periods in the Quick Reference Guide.

Time series for all outgoing traffic on S0:
rwfilter --sensor=S0 --type=out,outweb \ 
   --start=2009/04/21 --end=2009/04/23 \ 
   --proto=0- --pass=stdout \ 
| rwcount --bin-size=$((24*60*60))
rwcount Exercise

Produce a time-series with 30-minute intervals, analyzing incoming ICMP traffic collected at sensor S0 on April 21, 2009.
### rwcount Exercise solution

```bash
rwfilter --sensor=S0 --type=in,inicmp \[--start=2009/04/21 --proto=1 \[--pass=stdout \[
| rwcount --bin-size=1800
```

<table>
<thead>
<tr>
<th>Date</th>
<th>Records</th>
<th>Bytes</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009/04/21T13:00:00</td>
<td>5.00</td>
<td>960.00</td>
<td>16.00</td>
</tr>
<tr>
<td>2009/04/21T13:30:00</td>
<td>5.00</td>
<td>1500.00</td>
<td>25.00</td>
</tr>
<tr>
<td>2009/04/21T14:00:00</td>
<td>22.00</td>
<td>3900.00</td>
<td>65.00</td>
</tr>
<tr>
<td>2009/04/21T14:30:00</td>
<td>7.00</td>
<td>4136.00</td>
<td>42.00</td>
</tr>
<tr>
<td>2009/04/21T15:00:00</td>
<td>6.00</td>
<td>364.00</td>
<td>13.00</td>
</tr>
<tr>
<td>2009/04/21T15:30:00</td>
<td>16.00</td>
<td>1850.00</td>
<td>34.00</td>
</tr>
<tr>
<td>2009/04/21T16:00:00</td>
<td>8.00</td>
<td>934.00</td>
<td>19.00</td>
</tr>
</tbody>
</table>

...
rwcount --load-scheme

How are records, packets, and bytes allocated to flows that span time bins?

Time 1 bin  |  Time 2 bin  |  Time 3 bin

Flow

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Time 1 bin</th>
<th>Time 2 bin</th>
<th>Time 3 bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
rwcount --load-scheme

How do I choose a loading scheme? (hint: use default)

0 – Average load/bin (smooth peaks/valleys among bins)
1 – Flow onset / periodic behavior emphasis
2 – Emphasize flow termination
3 – Emphasize payload transfer above setup/termination

4 – Average load/time (smooth peaks/valleys over time)
5 – Worst case service loading
6 – Best case service loading

Most commonly used schemes are: 4, 0, 1
Calling `rwstats`

**rwstats --overall-stats**

- Descriptive statistics on byte and packet counts by record
- See “man rwstats” for details.

**rwstats --fields=KEY --value=VOLUME**

- `--count=N` or `--threshold=N` or `--percentage=N`
- `[--top]` or `[--bottom]`

- Choose one or two key fields.
- Count one of records, bytes, or packets.
- Great for Top-N lists and count thresholds
- (standard output formatting options – see “man rwstats”)
**What Is This? – 10**

```bash
rwfilter outtraffic.rw \
  --stime=2010/12/08T18:00:00-2010/12/08T18:59:59 \
  --pass=stdout \
  | rwstats --fields=sip --values=bytes --count=10
```

**INPUT:** 1085277 Records for 1104 Bins and 4224086177 Total Bytes

**OUTPUT:** Top 10 Bins by Bytes

<table>
<thead>
<tr>
<th>sip</th>
<th>bytes</th>
<th>%bytes</th>
<th>cumul_%</th>
</tr>
</thead>
<tbody>
<tr>
<td>71.55.40.62</td>
<td>1754767148</td>
<td>41.541935</td>
<td>41.541935</td>
</tr>
<tr>
<td>71.55.40.169</td>
<td>1192063164</td>
<td>28.220617</td>
<td>69.762552</td>
</tr>
<tr>
<td>71.55.40.179</td>
<td>331310772</td>
<td>7.843372</td>
<td>77.605923</td>
</tr>
<tr>
<td>71.55.40.204</td>
<td>170966278</td>
<td>4.047415</td>
<td>81.653338</td>
</tr>
<tr>
<td>177.249.19.217</td>
<td>122975880</td>
<td>2.911301</td>
<td>84.564639</td>
</tr>
<tr>
<td>71.55.40.72</td>
<td>110726717</td>
<td>2.621318</td>
<td>87.185957</td>
</tr>
<tr>
<td>71.55.40.200</td>
<td>101593627</td>
<td>2.405103</td>
<td>89.591060</td>
</tr>
<tr>
<td>177.71.129.255</td>
<td>40166574</td>
<td>0.950894</td>
<td>90.541954</td>
</tr>
<tr>
<td>71.55.40.91</td>
<td>35316554</td>
<td>0.836076</td>
<td>91.378030</td>
</tr>
<tr>
<td>149.249.114.204</td>
<td>26634602</td>
<td>0.630541</td>
<td>92.008571</td>
</tr>
</tbody>
</table>
rwstats Exercise 1

What are the top 10 incoming protocols on April 22, 2009, collected on sensor S0?
rwstats Exercise 1 solution

```
rxfilter --sensor=S0 --type=in,inweb \  
   --start=2009/04/22 --prot=0- --pass=stdout \  
| rxstats --fields=protocol --value=rec --count=10

INPUT: 337595 Records for 4 Bins and 337595 Total Records

OUTPUT: Top 10 Bins by Records

<table>
<thead>
<tr>
<th>pro</th>
<th>Records</th>
<th>%Records</th>
<th>cumul_%</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>336037</td>
<td>99.5385</td>
<td>99.5385</td>
</tr>
<tr>
<td>17</td>
<td>1467</td>
<td>0.4345</td>
<td>99.9730</td>
</tr>
<tr>
<td>1</td>
<td>88</td>
<td>0.0261</td>
<td>99.9991</td>
</tr>
<tr>
<td>132</td>
<td>3</td>
<td>0.0009</td>
<td>100.0000</td>
</tr>
</tbody>
</table>
```
rwstats Exercise 2

Top 10 inside hosts according to how many outside hosts they communicate with.

Use `--value=distinct:dip`
**rwstats Exercise 2 solution**

```bash
rwfilter --sensor=S0 --type=out,outweb --proto=0- \ 
 --start-d=2009/4/22 --pass=stdout \ 
| rwstats --fields=sip --value=distinct:dip --count=10
```

**INPUT:** 313028 Records for 7 Bins

**OUTPUT:** Top 10 Bins by dIP-Distinct

<table>
<thead>
<tr>
<th>sIP</th>
<th>dIP-Distinct</th>
<th>%dIP-Distinct</th>
<th>cumul_%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.60.187</td>
<td>50</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.5</td>
<td>26</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.25</td>
<td>17</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.73</td>
<td>14</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.191</td>
<td>11</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.251</td>
<td>9</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>10.1.60.132</td>
<td>3</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

--no-percents will clean up the question marks.
rwuniq

Unlike rwstats, rwuniq will display all the bins, not just the top or bottom N bins.

Output is normally unsorted. --sort-output causes sorting by the key (bin), unlike rwstats which sorts by aggregate value.
Calling `rwuniq`

`rwuniq --fields=KEY --value=VOLUME`

- Choose one or several key fields.
- Aggregate volume count: records, bytes, or packets.
- (standard output formatting options – see “man rwuniq”)

Apply thresholds to bins before outputting:

- `--bytes`, `--packets`, `--flows`, `--sip-distinct`, `--dip-distinct`
- Specify minimum aggregate value or a range

`--sort-output` by key (`rwstats` sorts by value)
rwfilter outtraffic.rw \
   --stime=2010/12/08:18:00:00-2010/12/08:18:59:59 \
   --saddress=71.55.40.62 --pass=stdout \
| rwuniq --fields=dip,sport --all-counts --sort-output

<table>
<thead>
<tr>
<th>dIP</th>
<th>sPort</th>
<th>Bytes</th>
<th>Packets</th>
<th>Records</th>
<th>sTime-Earliest</th>
<th>eTime-Latest</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.113.41.190</td>
<td>80</td>
<td>12782</td>
<td>20</td>
<td></td>
<td>2010/12/08T18:42:51</td>
<td>2010/12/08T18:58:49</td>
</tr>
<tr>
<td>30.182.228.143</td>
<td>80</td>
<td>203907933</td>
<td>143611</td>
<td></td>
<td>2010/12/08T18:53:59</td>
<td>2010/12/08T19:01:47</td>
</tr>
<tr>
<td>82.180.203.87</td>
<td>80</td>
<td>213013145</td>
<td>150896</td>
<td></td>
<td>2010/12/08T18:06:36</td>
<td>2010/12/08T18:32:33</td>
</tr>
<tr>
<td>82.180.203.197</td>
<td>80</td>
<td>800</td>
<td>8</td>
<td></td>
<td>2010/12/08T18:43:30</td>
<td>2010/12/08T18:43:30</td>
</tr>
<tr>
<td>88.124.166.233</td>
<td>80</td>
<td>223930369</td>
<td>158276</td>
<td></td>
<td>2010/12/08T18:08:55</td>
<td>2010/12/08T18:32:25</td>
</tr>
<tr>
<td>88.124.166.233</td>
<td>443</td>
<td>509285</td>
<td>732</td>
<td></td>
<td>2010/12/08T18:06:57</td>
<td>2010/12/08T18:51:11</td>
</tr>
<tr>
<td>94.239.226.247</td>
<td>80</td>
<td>124833037</td>
<td>96047</td>
<td></td>
<td>2010/12/08T18:25:22</td>
<td>2010/12/08T19:21:34</td>
</tr>
<tr>
<td>109.95.61.80</td>
<td>80</td>
<td>8467397</td>
<td>6325</td>
<td></td>
<td>2010/12/08T18:08:59</td>
<td>2010/12/08T18:10:09</td>
</tr>
<tr>
<td>139.65.186.4</td>
<td>80</td>
<td>204123360</td>
<td>143794</td>
<td></td>
<td>2010/12/08T18:19:48</td>
<td>2010/12/08T18:26:36</td>
</tr>
<tr>
<td>139.177.10.136</td>
<td>80</td>
<td>407978375</td>
<td>287354</td>
<td></td>
<td>2010/12/08T18:20:03</td>
<td>2010/12/08T19:01:30</td>
</tr>
<tr>
<td>219.149.72.154</td>
<td>1024</td>
<td>44</td>
<td>1</td>
<td></td>
<td>2010/12/08T18:50:40</td>
<td>2010/12/08T18:50:40</td>
</tr>
<tr>
<td>249.216.88.172</td>
<td>80</td>
<td>88</td>
<td>2</td>
<td></td>
<td>2010/12/08T18:44:42</td>
<td>2010/12/08T18:44:47</td>
</tr>
<tr>
<td>250.211.100.88</td>
<td>80</td>
<td>3295160</td>
<td>2492</td>
<td></td>
<td>2010/12/08T18:47:50</td>
<td>2010/12/08T18:58:53</td>
</tr>
</tbody>
</table>
What Is This? – 12

rwuniq outtraffic.rw --fields=dip \ 
--values=sip-distinct,records,bytes --sip-distinct=400- \ 
--sort-output

<table>
<thead>
<tr>
<th>dIP</th>
<th>SIP-Distinct</th>
<th>Bytes</th>
<th>Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.220.28.183</td>
<td>512</td>
<td>20480</td>
<td>512</td>
</tr>
<tr>
<td>171.128.2.27</td>
<td>448</td>
<td>19069280</td>
<td>476732</td>
</tr>
<tr>
<td>171.128.2.179</td>
<td>448</td>
<td>139501200</td>
<td>3487530</td>
</tr>
<tr>
<td>171.128.212.14</td>
<td>448</td>
<td>139467440</td>
<td>3486686</td>
</tr>
<tr>
<td>171.128.212.124</td>
<td>448</td>
<td>127664480</td>
<td>3191612</td>
</tr>
<tr>
<td>171.128.212.127</td>
<td>448</td>
<td>66611560</td>
<td>1665289</td>
</tr>
<tr>
<td>171.128.212.188</td>
<td>448</td>
<td>139467680</td>
<td>3486692</td>
</tr>
<tr>
<td>171.128.212.228</td>
<td>448</td>
<td>139393160</td>
<td>3484829</td>
</tr>
<tr>
<td>245.225.153.120</td>
<td>763</td>
<td>30520</td>
<td>763</td>
</tr>
<tr>
<td>245.238.193.102</td>
<td>1339</td>
<td>179480</td>
<td>4487</td>
</tr>
</tbody>
</table>
## rwuniq vs. rwstats

<table>
<thead>
<tr>
<th>rwuniq</th>
<th>both</th>
<th>rwstats in top/bottom mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>all bins except per thresholds</td>
<td>Bin by key</td>
<td>--top or --bottom bins</td>
</tr>
<tr>
<td>--sort-output by key</td>
<td>Default aggregate value is flows</td>
<td></td>
</tr>
<tr>
<td>otherwise unsorted</td>
<td>(records)</td>
<td></td>
</tr>
<tr>
<td>Thresholds or ranges: --bytes,</td>
<td>Choose which bins have aggregate</td>
<td>--count, --threshold,</td>
</tr>
<tr>
<td>--packets, --flows, --sip-distinct,</td>
<td>values significant enough to output.</td>
<td>--percentage</td>
</tr>
<tr>
<td>--dip-distinct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--all-counts (bytes, pkts, flows, earliest</td>
<td>Show volume aggregate value[s]</td>
<td>--no-percents (good when</td>
</tr>
<tr>
<td>sTime, and latest eTime)</td>
<td></td>
<td>primary aggregate isn’t</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes, Packets, or Records)</td>
</tr>
<tr>
<td>--bin-time to adjust sTime and eTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--presorted-input (omit when value includes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distinct fields, even if input is sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--values=sTime-Earliest, eTime-Latest</td>
<td>--values=Records, Packets, Bytes,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sIP-Distinct, dIP-Distinct,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distinct:KEY-FIELD (KEY-FIELD can’t</td>
<td></td>
</tr>
<tr>
<td></td>
<td>also be key field in --fields)</td>
<td></td>
</tr>
</tbody>
</table>
Blacklists, Whitelists, Books of Lists...

Too many addresses for the command line?

- spam block list
- malicious websites
- arbitrary list of any type of addresses

Create an IP set!

- individual IP address in dotted decimal or integer
- CIDR blocks, 192.168.0.0/16
- wildcards, 10.4,6.x.2-254

Use it directly within your filter commands.

- --sipset, --dipset, --anyset
Set Tools

**rwsetbuild**: Create sets from text.
**rwset**: Create sets from binary flow records.
**rwsetcat**: Display an IP set as text.
**rwsetmember**: Test if an address is in given IP sets.
**rwsettool**: Perform set algebra (intersection, union, set difference) on multiple IP sets.
Set Intersection

rwsettool --intersect web.set dns.set --output web_and_dns.set
Set Union

Web Servers  DNS Servers

Hosts that are either Web servers, DNS servers, or both

rwsettool --union web.set dns.set --output web_or_dns.set
Set Difference

Web Servers

DNS Servers

Hosts that are Web servers, but not DNS servers

rwsettool --difference web.set dns.set --output web_not_dns.set
more MSSP.txt
171.128.2.0/24
171.128.212.0/24

rwsetbuild MSSP.txt MSSP.set
rwfilter --start=2010/12/8 --anyset=MSSP.set \ 
   --pass=MSSP.rw --print-vol

<table>
<thead>
<tr>
<th>Recs</th>
<th>Packets</th>
<th>Bytes</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>30767188</td>
<td>81382782</td>
<td>35478407950</td>
</tr>
<tr>
<td>Pass</td>
<td>26678669</td>
<td>31743084</td>
<td>1464964676</td>
</tr>
<tr>
<td>Fail</td>
<td>4088519</td>
<td>49639698</td>
<td>34013443274</td>
</tr>
</tbody>
</table>

rwset --sip-file=MSSPsource.set MSSP.rw
rwsettool --intersect MSSP.set MSSPsource.set \ 
   --output=activeMSSP.set
rwsetcat --count-ips activeMSSP.set
22
rwfilter --sensor=S0 --type=out \ 
   --start=2009/4/21 --proto=0- \ 
   --pass=stdout \ 
   | rwset --dip-file=outIPs.set

rwsetcat outIPs.set --network-structure=16

  10.1.0.0/16| 8748
  10.2.0.0/16| 27
  140.13.0.0/16| 1
Set Exercise 1

Make a set-file of addresses of all actual inside hosts. Should we examine incoming or outgoing traffic? Make a set-file of all outside addresses. Can you make both sets with one command?
Set Exercise 1 solution

rwfilter --sensor=S0 --type=out,outweb \ 
  --start-d=2009/4/21 --end=2009/4/23 \ 
  --proto=0- --pass=stdout \ 
| rwset --sip-file=insidehosts.set \ 
  --dip-file=outsidehosts.set
Set Exercise 2

Examine the two set-files.
Set Exercise 2 solution

ls -l insidehosts.set
rwfileinfo insidehosts.set
rwsetcat insidehosts.set

ls -l outsidehosts.set
rwsetcat outsidehosts.set | less
Set Exercise 3

Which /24 networks are on the inside?
Which /24 networks are on the outside?
Set Exercise 3 solution

rwsetcat --network-struc=24 insidehosts.set
rwsetcat --network-struc=24 outsidehosts.set
Advanced Partitioning Options

- TCP Flags
- Count of packets and bytes
- Time
- Extending rwfilter’s partitioning options with plugins
TCP Flags

S – Syn (synchronize)
U – Urg (urgent)
R – Rst (reset)
F – Fin (finish)
P – Psh (push)
A – Ack (acknowledge)
C – CWR (congestion window reduced)
E – ECE (explicit congestion notification echo)
TCP Flags

--flags-initial= # TCP flags in 1\textsuperscript{st} pkt of flow
--flags-session= # flags in remaining packets
--flags-all= # flags in all pkts of flow

=flagsOn/flagsExamined

flagsOn: TCP flags that must be On to pass.
flagsExamined: flags under consideration for passing.
Any flags in flagsOn must also be in flagsExamined.
Flags in flagsExamined, but not in flagsOn, must be off to pass.
TCP Flags

--flags-initial=S/SA # flow from client to server
--flags-initial-SA/SA # flow from server to client
--flags-init=S/SA --flags-session=F/F # full C->S flow
--flags-init=SA/SA --flags-session=F/F # full S->C flow
--flags-all=S/SFR # incomplete flow
--flags-all=/FR # unfinished flow fragment
Count of Packets and Bytes

--packets= # packets in the flow
--bytes= # bytes in the packets in flow
--bytes-per-packet= # average

--packets=3-
--bytes=40-570
--bytes-per-packet=40.0-75.125
Partitioning by Time

--stime=earliertime-latertime
--etime=earliertime-latertime
--active-time=earliertime-latertime
--duration=lowseconds-highseconds

stime and etime are usually not used together.
Each time has millisecond resolution.

--etime=2009/4/21T13:00:00-2009/4/21T13:00:09 # 10 sec
Extending Partitioning with Plugins

rwfilter’s partitioning capabilities can be extended with plugins written in Python or C.

--python-expr= # simple python expression
--python-file= # complex python pgm in a file
--plugin= # compiled C program in a file

--python-expr='rec.sport == rec.dport'
--python-file=clientserver_filt.py
--plugin=app-mismatch.so
I Only Believe What I See

You’ll be tempted to work with text-based records.

- It’s easy to see the results and post-process with other tools (e.g., Perl, awk, sed, sort).
- It takes a lot of space, and it’s much, much slower.

Guiding principle: Keep flows in binary format as long as possible.
rwfilter --type=out \\ --start=2010/12/08 \\ --aport=22 --pass=ssh.rw

rwfilter --dport=22 ssh.rw \\ --pass=stdout | rwcut

rwfilter --sport=22 ssh.rw \\ --pass=stdout | rwcut
Outline — 4

Introduction: SiLK
Network flow
Basic SiLK tools
Advanced SiLK tools
Summary
PySiLK—Using SiLK with Python

- PySiLK—an extension to Python
- Allows Python to manipulate SiLK’s data files
- Uses the “silk” python module, from SEI CERT.
PySiLK components

PySiLK
- Read, manipulate, and write SiLK Flow records, IPsets, Bags, and Prefix Maps (pmaps) from within Python

SilkPython (--python-file=)
- Create plug-ins for rwfilter or other SiLK utilities.
  - Create partitioning switches for rwfilter
  - Create new flow-record fields for other utilities

--python-expr=
- Create a simple partitioning test without creating a new switch
Stand-alone PySiLK example

```python
#!/usr/bin/env python
import silk
myfile = silk.silkfile_open("MyFlows.rw", silk.READ)
for rec in myfile:
    if rec.sport < 2500 and rec.sport == rec.dport:
        print rec.sport, rec.stime, rec.sip, rec.dip
myfile.close()
```
PySiLK exercise

Write a Python program which reports the source IP address associated with the lowest source port used by any flow record in the file MyFlows.rw.
#! /usr/bin/env python

import silk

lowsport = 65536  # could use 99999
myfile = silk.silkfile_open("MyFlows.rw", silk.READ)

for rec in myfile:
    if rec.sport < lowsport:
        lowsport = rec.sport
        lowsip = rec.sip

myfile.close()

print rec.sport, rec.sip
--python-expr example

rwfilter sample.rw \  
  --protocol=6 \  
  --python-expr='rec.sport == rec.dport' \  
  --pass=equalTCPports.rw
import silk

def lowerport(rec):
    if rec.sport < rec.dport:
        return rec.sport
    else:
        return rec.dport

register_int_field('lport', lowerport, 0, 65535)
SilkPython example (2)

```
rwstats --python-file=lowport.py --fields=lport \ 
   --value=records --count=10 flows.rw
```
SilkPython exercise

Write a plug-in for rwcut, rwstats, etc. The plug-in should define a new flow-record field which contains the IP address of the host using the lower port number in the flow. You’ll need the following SilkPython function:

```python
register_ip_field(field_name, ip_function)
```
import silk

def lowerport_ip(rec):
    if rec.sport < rec.dport:
        return rec.sip
    else:
        return rec.dip

register_ip_field("lip", lowerport_ip)
Alternatives to PySiLK

• SiLK tools
  • Not as flexible criteria as Python.
  • Could use tuple files
    • Must be maintained
    • Aren’t self-contained with logic
    • Large tuple files run slower than Python.

• Text processing with Perl, C, or Java
  • Create text with rwcut delimited without titles
  • Convert ports back to integers
  • Dealing with dates, times, or addresses difficult
Modified example of PySilk

- Summarize the selection as a count by port
- Just keep a Python dictionary
  - Key = port number
  - Value = count
PySiLK advantages

- Speeds both programming and processing
  - Keeps data in binary, unlike Perl & C
    - No parsing text
  - Built-in conversions of objects to strings
  - Full power of Python
- Good for:
  - Stateful filters and output options
  - Integrate SiLK with other data types
  - Complex or branching filter rules
  - Custom key fields and aggregators for rwcut, rwsort
Outline — 5

Introduction: SiLK
Network flow
Basic SiLK tools
Advanced SiLK tools
Summary
Furthering Your SiLK Analysis Skills (1)

Each tool has a `--help` option.

SiLK Reference Guide
SiLK Analysts’ Handbook
  - Both available at the SiLK tools website
    [http://tools.netsa.cert.org](http://tools.netsa.cert.org)

Email support
  - [silk-help@cert.org](mailto:silk-help@cert.org)
Furthering Your SiLK Analysis Skills (2)

Tool tips

- [SiLK Tooltips link on http://tools.netsa.cert.org](http://tools.netsa.cert.org)

Flow analysis research and advanced techniques

- [http://www.cert.org/flocon](http://www.cert.org/flocon)
- [http://www.cert.org/netsa](http://www.cert.org/netsa)
Questions?
Contact Information

Ron Bandes — rbandes@cert.org
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA