SiLK Tool Suite Quick Reference

Binary Output SiLK Tools

(Tools without page numbers are not in this guide.)
(See back cover for a list of text output tools.)

Key (colors/letters)

Black tools produce flow binary.
Green tools produce bag binary. (B)
Blue tools produce pcap binary. (P)
Purple tools produce IPset binary. (I)
Orange tools produce other binary formats. (O)

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

SiLK Flow Record Fields

NUM	NAME	DESCRIPTION
1	sip	Source IP address
2	dip	Destination IP address
3	sport	Source port
4	dport	Destination port *
5	proto	Protocol value in IP header
6	packets	Packet count
7	bytes	Byte count
8	flags	TCP flags from all packets
9	stime	Start time
10	dur	Duration
11	etime	End time
12	sensor	Sensor name/ID
13	in	(Unused)
14	out	(Unused)
15	nhip	(Used for marking)
16	stype	Index from address_types.pmap for sip
17	dtype	Index from address_types.pmap for dip
18	scc	Country code of sip
19	dcc	Country code of dip
20	class	Sensor category
21	type	Flow category (in, out, inweb, outweb, etc.)
22	stime+msec	Start time forcing milliseconds
23	dur+msec	Duration forcing milliseconds
24	etime+msec	End time forcing milliseconds
25	icmptypecode	ICMP type & code
26	initialflags	TCP flags for first packet
27	sessionflags	TCP flags for later packets
28	attributes	Termination conditions
29	application	Service recognition
	itype	ICMP type *
	icode	ICMP code *
	src-*mapname*	Label for sip or proto/port from mapname
	dst-*mapname*	Label for dip or proto/port from mapname

* *ICMP itype and icode are multiplexed and stored in the destination port field.*

SiLK Parameter Formats

Parameter order is up to the user except that parameters created via pmaps and plugins must be defined before they are referenced.

General Parameter Formats

`--name=argument`

Where name may be shortened to the minimum prefix not shared with another parameter (e.g. ”protocol” can be ”pro” but not ”pr” for `rwfilter`)

`filename`

Where name follows Linux path formats, or may be `stdin` or `stdout` (as appropriate), or named pipe

Argument Formats

Attr-mask	*High/Care,* where both *High* and *Care* are a series of `FTCS`
Cc-list	Comma-separated list of top-level country code abbreviations
Cidr-list	Comma-separated list of IP addresses (in dotted-decimal notation)
Date	YYYY/MM/DD`T`HH or YYYY/MM/DD
Decimal	*Decimal–Decimal* or *Decimal–*
Fieldlist	Comma-separated list of field names or *Int-range*
Flag-mask	TCP flags as *High/Care* or comma-separated list of *High/Care* where both *High* and *Care* are a series (no separator) of `SFARPECU`
Integer	Any positive whole number, range specified by context (e.g. 123 or aaa9)
Int-range	*Integer–Integer* or *Integer*
Int-list	Comma-separated list of *Int-range* or *Integer*
Ip-addr	A CIDR block or a single IPv4 or IPv6 in canonical or integer notation; any field of a canonical address can be an *int-list* or the wild card `x`
Sensors	Comma-separated list of sensor names or *Int-range*
String	Sequence of characters between quotes
Time	YYYY/MM/DDTHH or YYYY/MM/DDTHH:MM or YYYY/MM/DDTHH:MM:SS or YYYY/MM/DDTHH:MM:SS.mmm (T can be replaced by :)
Time-range	*Time–Time*

Compression Options (Comp. Opt.)

none	No compression
zlib	Best compression, slower performance
lzo1x	Lesser compression, better performance (default)
best	Implementation defined (currently *lzo1x*)

rwfilter

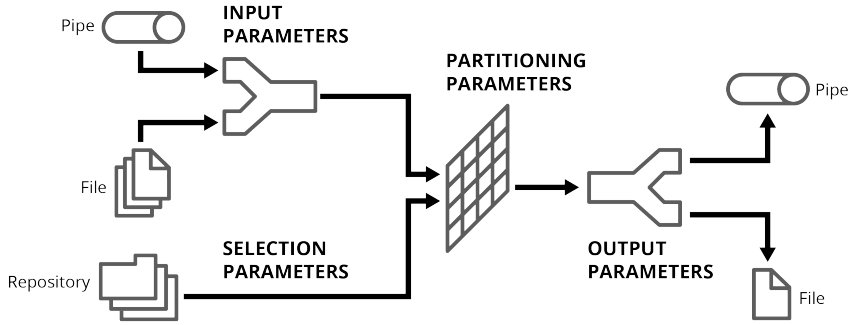
Retrieve flow records from pipe, file, or repository; select records of interest; and store to pipe or file.

Syntax Summary

`rwfilter` *`input selection partitioning output other`*

Input or selection [not both], partitioning, and output are required

How rwfilter Works *



** Other parameters affect the general function of the tool or provide informative output to the user*

Examples

Pull outbound traffic to address block for 8 hours:

```
rwfilter --start-date=2018/04/15T00 --end=2018/04/15T07 \
 --sensor=SEN1 --type=out --daddress=10.5.x.x --pass=10-5.rw
```

Pull all inbound traffic for 15 minutes:

```
rwfilter --start=2018/04/15T00 --sensor=SEN1 \
 --stime=2018/04/15T00:00-2018/04/15T00:15 --pass=first-qtr.rw
```

Choose completed TCP flows with content from a file:

```
rwfilter all-outbound.rw --proto=6 --flags-all=SAF/SAF,SAR/SAR \
 --packets=4- --bytes-per=65- --pass=comp-tcp.rw
```

For one hour of traffic, split common protocols from less common into two files:

```
rwfilter --start=2018/04/15T00 --sensor=SEN1 --type=all \
 --proto=1,6,17,50,51 --pass=common.rw --fail=less_common.rw
```

`rwfilter` Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Input Parameters	
data	Root directory of repository	dirname
	Flow file/pipe (no parameter prefix, multiple allowed)	
site-conf	Location of site configuration file	filename
xarg	Read input file names from file/pipe	filename (opt.)
	Selection Parameters	
class	Class of sensor to process	string
end	Final hour of data to process	string
flowtype	Class/type pairs to process	class/type
sensor	Sensors to process	sensors
start	First hour of data to process	date
type	Types of flow records to process	type
	Output Parameters	
all	Destination for all records	filename/pipe/`stdout`
fail	Destination for records that fail	filename/pipe/`stdout`
pass	Destination for records that pass	filename/pipe/`stdout`
print-miss	Print the names of missing files	none
print-stat	Print a count of total flows	filename (opt.)
print-vol	Print count of flows/packets/bytes	filename (opt.)

Start-date and End-date

		--start-date		
		HOUR	DAY	NONE
--end-date	HOUR	HOURS IN EXPLICIT RANGE	IGNORE end-date HOUR. WHOLE DAYS. (DON'T DO THIS)	ERROR
	DAY	end-hour IS THE SAME AS start-hour. # HOURS = 1, 25, 49, ... (DON'T DO THIS)	WHOLE DAYS	ERROR
	NONE	1 HOUR	1 DAY	CURRENT DAY TO PRESENT TIME.

`rwfilter` **Parameters** *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
	Other Parameters	
compress	Set compression for output	comp. opt.
dry-run	Report command line errors	none
help	Print command summary	none
max-fail	Write at most *arg* records to fail	integer (0=all)
max-pass	Write at most *arg* records to pass	integer (0=all)
note-add	Put *arg* in file header	string
note-file	Put content of *arg* in file header	filename
plugin	Use plugin to filter records	filename
print-file	Print names of input files	none
thread	Set filter threads	integer
version	Print version	none
	Partitioning Parameters	
	also class, flowtype, type, sensor from selection	
active	Flow active during this time window	time-range
any-addr	Source or destination address matches IP (and not–)	ip-addr
any-cc	Source or destination country code	country code list
any-cidr	Source or destination address in this list (and not–)	cidr-list
anyset	Source or destination address in this IPset (and not–)	filename
aport	Source or destination port in this list	int-list
app	Flow signature label in list	int-list
attrib	Attribute field matches list	attr-mask
bytes	Byte count within this range	int-range
bytes-per	Byte-per-packet count within this range	dec-range
daddr	Destination address matches IP (and not–)	ip-addr
dcc	address maps country code in list	cc-list
dcidr	Destination address in this list (and not–)	cidr-list
dipset	Destination address in this IPset (and not–)	filename
dport	Destination port in this list	int-list
dtype	Destination address index in address_types.pmap matches *arg*	integer
dur	Duration in seconds within this range	dec-range
etime	Ending time within this time window	time-range
flags-all	TCP flags match list	flags-mask
flags-init	Initial TCP flags match list	flags-mask
flags-sess	Session TCP flags match list	flags-mask
icmp-code	ICMP or ICMPv6 code is in this list	int-list

`rwfilter` **Parameters** *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
	Partitioning Parameters (continued)	
icmp-type	ICMP or ICMPv6 type is in this list	int-list
ip-v	IP version in list	int-list
ipa-any	Source or destination address matches expression	string
ipa-dst	Destination address matches expression	string
ipa-s	Source address matches expression	string
next	nhIP field matches IPO (and not–)	ip-addr
nhcidr	nhIP field in this list (and not–)	cidr-list
nhipset	nhIP field in IP set (and not–)	filename
packet	Packet count within this range	int-range
pmap-any-MAPNAME	Source or destination address map labeling matches argument	string
pmap-dst-MAPNAME	Destination address map labeling matches argument	string
pmap-file	Prefix map file to read	filename
pmap-src-MAPNAME	Source address map labeling matches argument	string
proto	Protocol in this list	int-list
python-exp	Run expression	string
python-file	Use Python code to extend processing	filename
saddr	Source address matches IP (and not–)	ip-addr
scc	Source address maps country code in list	cc-list
scidr	Source address in this list (and not–)	filename
sipset	Source address in IP set (and not–)	filename
sport	Source port in this list	int-list
stime	Start time within this time window	time-range
stype	Source address index in address_types.pmap matches Arg	integer
tcp-flag	TCP flags are in the list	`FSRPAUEC`
tuple-del	Character separating the fields	character
tuple-dir	Specify IP-port mapping `forward`—as given `reverse`—flip source and destination `both`—do either matching	
tuple-field	Fields in five-tuple with values in tuple file	fieldlist
tuple-file	Record five-tuple fields match value combinations in file	filename

rwcut

Display network flow records as columnar or delimited text.

Syntax Summary

`rwcut` *`formatting-parameters range-parameters output-parameters filename ...`*

File or pipe from `stdin`; default fields are 1-12.

Examples

Quick overview of records in file:

```
rwcut --fields=1-6,stime flows.rw --pager=less
```

Output full records from file in csv format (`sed` command adds space after each comma):

```
rwcut --all-fields --delim=',' flows.rw \
| sed -e 's/,/, /g' >flows.csv
```

Output data with integer IP addresses (rather than dotted-quad) for sorting, plotting etc.:

```
rwcut --ip-format=decimal --fields=sip,dip \
flows.rw >flows.txt
```

Changing order of columnar display:

```
rwcut --fields=protocol,sip,sport,dip,dport \
flows.rw >flows.txt
```

Labeling source addresses using a pmap:

```
rwcut --pmap-file=mal:malware.pmap --pmap-col=10 \
--fields=src-mal,1-7,stime flows.rw >mal-flows.txt
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Output Parameters	
copy	Copy all input SiLK flows to given pipe or file	filename
dry-run	Parse options and print column titles only	none
help	Print command summary	none
help-fields	Print field descriptions	none
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
print-file	Print names of input files as they are opened	none
site-conf	Specify location of the site configuration file	filename
version	Print program version	none

`rwcut` **Parameters** *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
	Range Parameters	
all-fields	Print all known fields to the output	none
end-rec	Specify ending record number	integer
fields	Specify fields to print	fieldlist
ipv6-policy	Specify how to handle IPv4 and IPv6 records `ignore`—drop IPv6 records `asv4`—convert v6 to v4 else ignore `mix`—allow both `force`—convert v4 to v6 `only`—drop IPv4	
num	Specify number of records to print	integer
pmap-file	Prefix map file to read	map:filename
start-rec	Specify starting record number	integer
tail-recs	Specify starting record number from end of file	integer
xarg	Read input filenames from file or pipe	filename (opt.)
	Formatting Parameters	
col	Specify separation character between columns	character
delim	Shortcut for no-columns no-final-del column-sep	character (opt.)
icmp	Print ICMP type & code in sPort and dPort fields	none
integer-sen	Print sensor as an integer	none
integer-tcp	Print TCP flags as an integer	none
ip-format	Specify IP address print format `canonical`—dotted quad (IPv4) or hexadectet (IPv6) `decimal`—integers `force ipv6`—print all addresses as IPv6 `hexadecimal`—base-16 integers `zero-pad`—add zeroes to fully fill column	
no-col	Disable fixed-width columnar output	none
no-final	Suppress column delimiter after last field	none
no-titles	Do not print column headers	none

`rwcut` **Parameters** *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
	Formatting Parameters	
plugin	Load given plugin to add fields	filename
pmap-col	Maximum column width to use for pmap value output	integer
python-file	Use Python code to extend processing	filename
timestamp	Time format options `default`—yyyy/mm/ddThh:mm:ss.sss `iso`—yyyy-mm-dd hh:mm:ss.sss `m/d/y`—mm/dd/yyyy hh:mm:ss.sss `epoch`—seconds since epoch (ignores tz) `utc`—use UTC timezone `local`—use local timezone `no-msec`—truncate milliseconds	

Formatted Output

```
$ rwcut --fields=1-7 --num-recs=3 web.rw
            sIP|            dIP|sPort|dPort|pro|   packets|     bytes|
     10.0.40.21|192.168.164.227|  443|62396|  6|        13|      2157|
192.168.164.227|     10.0.40.21|62396|  443|  6|        20|      5990|
192.168.163.105|     10.0.40.21|54883|  443|  6|         8|       344|

$ rwcut --fields=1-7 --num-recs=3 --delim=, web.rw
sIP,dIP,sPort,dPort,protocol,packets,bytes
10.0.40.21,192.168.164.227,443,62396,6,13,2157
192.168.164.227,10.0.40.21,62396,443,6,20,5990
192.168.163.105,10.0.40.21,54883,443,6,8,344

$ rwcut –fields=sip,sport,dip,dport --no-title --delim=, --num-recs=3 web.rw
10.0.40.21,443,192.168.164.227,62396
192.168.164.227,62396,10.0.40.21,443
192.168.163.105,54883,10.0.40.21,443
```

Padded Fields

Bar-delimited

No Padding, comma delimited

No Column headings

No Padding, comma delimited

Select Columns in specified order

rwfileinfo

Print summary information about SiLK binary format files (flow, set,bag, etc.).

Syntax Summary

`rwfileinfo` *`parameters filename ...`*
All non-file parameters are optional

Examples

Show all summary information on two files:

```
rwfileinfo flows.rw internal-ip.set
```

Show how file generated and any comments:

```
rwfileinfo --fields=command-lines,annotations flows.rw
```

Parameters

MIN-NAME	DESCRIPTION		ARGUMENTS
fields	List of fields to print		fieldlist
	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		
help	Print command summary		none
help-fields	Print field descriptions		none
no-titles	Suppress file names and field names		none
site-conf	Specify location of the site configuration file		filename
summary	Print total files; file sizes; records		none
version	Print this program's version		none
xarg	Read input file names from pipe or file		filename (opt.)

Formatted Output

```
$ rwfileinfo --fields=count,command web.rw
web.rw:
  count-records       3125624
  command-lines
                    1 rwfilter --start=2015/06/10 --end=2015/06/20
--type=inweb,outweb --proto=6 --aport=80,8080,443 --packets=8- --pass=stdout
                    2 rwsort --fields=stime
```

rwsiteinfo

Displays information about site collection configuration, including sensor names and numbers. (Replaces `mapsid` command from prior versions of SiLK.)

Syntax Summary

`rwsiteinfo` *`parameters`* `--fields=`*`site-fields`*

`--fields` *is required*

Examples

Print list of all sensor names and numbers:

```
rwsiteinfo --fields=sensor,id-sensor
```

Print sensor name for two sensor numbers:

```
rwsiteinfo --fields=sensor --sensor=0,1
```

Print description of a sensor:

```
rwsiteinfo --fields=describe-sensor --sensor=SEN1
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
classes	Display listed classes	string
col	Specify separation characters between columns	character
data	Root of directory containing repository	filename
delim	Shortcut for no-columns no-final-del column-sep	character (opt.)
fields	List of fields to print	site-fields
flowtypes	Display listed class/type pairs	class/type
help	Print command summary	none
help-fields	Print field descriptions	none
output	Send output to given file path	filename
list-delim	Use specified character in fields list	character
no-col	Disable fixed-width columnar output	filename
no-final	Suppress column delimiter after last	none
no-titles	Do not print column headers	none
pager	Program to invoke to process output	filename
sensors	Display listed sensors	int-list or name list
site-conf	Specify location of the site configuration file	filename

`rwsiteinfo` *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
timestamp-format	Specify formatting of times	(see `rwcut` description)
types	Display listed types	type list
version	Print this program's version	none

Site Fields

FIELD	DESCRIPTION
`class`	role of sensor as configured *
`flowtype`	class/type pair *
`default-class`	default sensor role *
`id-flowtype`	integer class/type pair *
`mark-defaults`	indicate use of defaults
`id-sensor`	integer sensor ID *
`default-type`	default flow category *
`sensor`	name of sensor *
`describe-sensor`	text description of sensor
`type`	flow category *
`repo-start-date`	time of first file
`repo-end-date`	time of latest file
`repo-file-count`	number of files

* *These fields also have a :list form (e.g. class:list) that formats the entry as a comma-separated list instead of across multiple lines.*

Formatted Output

```
$ rwsiteinfo --fields=type,mark-defaults,repo-start,repo-end
   Type|Defaults|          Start-Date|            End-Date|
     In|       +*|2015/06/02T13:00:00|2015/06/18T18:00:00|
    out|       + |2015/06/02T13:00:00|2015/06/18T18:00:00|
  inweb|       +*|2015/06/02T13:00:00|2015/06/18T18:00:00|
 outweb|       + |2015/06/02T13:00:00|2015/06/18T18:00:00|
int2int|       + |2015/06/02T13:00:00|2015/06/18T18:00:00|
ext2ext|       + |2015/06/02T13:00:00|2015/06/18T18:00:00|
```

Flow type available

* If default for rwfilter

Date/time of first record

Date/time of last record

rwset

Read binary flow records and generate one or more IPsets.

Syntax Summary

`rwset` *`option-parameters field-parameters source ...`*
Option-parameters and source are optional

Examples

Generate sets from source and destination IP addresses or records in file:

```
rwset --sip-file=src.set --dip-file=dst.set flows.rw
```

Generate sets with refiltering:

```
rwfilter --start=2011/04/15T00 --end=2011/04/15T07 \
--type=out --proto=6 --pass=stdout \
| rwset --sip=tcp-src.set --copy=stdout \
| rwfilter stdin --sport=0-1023 --pass=stdout \
| rwset --sip=tcp-rsvd-src.set
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Option Parameters	
compress	Select compression	comp. opt.
copy	Copy all input SiLK flows to given pipe or file	
help	Print command summary	none
invocation-strip	Remove command history from file header	none
ipv6-policy	Specify how to handle IPv4/v6 records	(see `rwcut` descr.)
note-add	Put arg in file header	string
note-file	Put contents of arg in file header	filename
note-strip	Remove note entries from file header	none
print-file	Print names of input files as they are opened	none
record-v	IPset record version for compatibility	`0`,`2`,`3`, or `4`
site-conf	Specify location of configuration file	filename

MIN-NAME	DESCRIPTION	ARGUMENTS
	Option Parameters	
version	Print program version	none
xarg	Read input file names from file or pipe	filename (opt.)
	Field Parameters (at least one needed)	
any-file	Store IPset of both source and destination addresses	filename
dip-file	Store IPset of destination addresses	filename
nhip-file	Store IPset of flow markings	filename
sip-file	Store IPset of source addresses	filename

rwsetbuild

Read text list of IP addresses and produce binary IPset.

Syntax Summary

`rwsetbuild` *`parameters input output`*
Can use `stdin` for input and `stdout` for output, otherwise filenames

Sample Input File

```
# list.set.txt - containing 10 addresses
10.1.1.1
10.2.2.2
192.168.12.0/29
```

Examples

Generate IPset from one-address-per-line file:

```
rwsetbuild list.set.txt list.set
```

Generate IPset from file with address ranges (colon-separated):

```
rwsetbuild --ip-range=':' ranges.set.txt ranges.set
```

Produce sorted list of unique IP addresses in file:

```
rwsetbuild input.txt stdout | rwsetcat
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
compress	Select compression	comp. opt.
help	Print command summary	none
invocation-strip	Remove command history from file header	none
ip-ranges	Allow input of address ranges in IP or integer format	
(no wildcards)	character (opt.)	
note-add	Put *arg* in file header	string
note-file	Put contents of *arg* in file header	filename
record-v	IPset record version for compatibility	`0`,`2`,`3`, or `4`
version	Print program version	none

rwsettool

Perform operations on IPset files to produce new IPset files.

Syntax Summary

`rwsettool` *`operation arg-sets parameters`*
Where `arg-sets` *is a blank-delimited list of IP set file names*

Examples

Merging two sets:

```
rwsettool --union day1.set day2.set --output=either.set
```

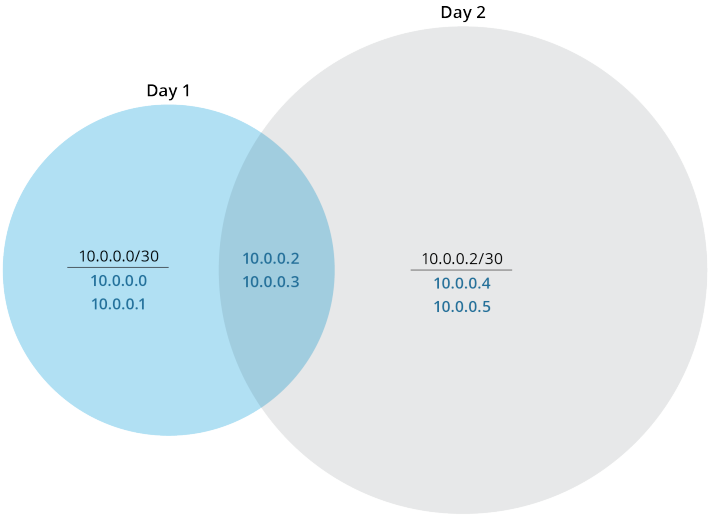
Finding common elements:

```
rwsettool --intersect day1.set day2.set --output=both.set
```

Finding non-common elements:

```
rwsettool --diff day1.set day2.set --output=only.set
rwsettool --diff day2.set day1.set \
| rwsettool --union stdin only.set --output=not-comm.set
```

Set Operations



Intersection = {10.0.0.2, 10.0.0.3}
Union = {10.0.0.0–10.0.0.5}
Difference = (Day 2 - Day 1) = {10.0.0.4, 10.0.0.5}
Non-common = {10.0.0.0, 10.0.0.1, 10.0.0.4, 10.0.0.5}

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

`rwsettool` Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Operations	
diff	Gathers IPs from first input not in other input IPsets	none
intersect	Gathers IPs that exist in ALL the input IPsets	none
mask	Only one IP per block of size in Arg	integer
sample	Only random samples from input; need size or ratio	none
symm	Gathers IPs that are unique to each input IPset	none
union	Gathers IPs that exist in ANY input IPset	none
	Option Parameters	
compress	Select compression	comp. opt.
fill	Use complete blocks of given prefix length	integer
help	Print command summary	none
invocation-strip	Remove command history from file header	none
note-add	Put arg in file header	string
note-file	Put contents of arg in file header	filename
note-strip	Remove note entries from file header	none
output	Specify output location	filename
ratio	The probability that an individual IP will be sampled	0.0-1.0
record-v	IPset record version for compatibility	`0`,`2`,`3`, or `4`
seed	The sample size for `--sample`	decimal
size	The sample size for `--sample` per input	integer
version	Print program version	none

rwsetcat

Read binary IPset and produce text.

Syntax Summary

`rwsetcat` *`parameters filename ...`*
All optional

Examples

List IP addresses in IPset into text file:

```
rwsetcat list.set >list.set.txt
```

Count IP addresses from standard input:

```
rwsetcat --count
```

Summarize CIDR /16 blocks (class B subnets) in IPset:

```
rwsetcat --net=16 list.set
```

List IP addresses in IPset as integers (for plotting):

```
rwsetcat --ip-format=decimal list.set
```

List IP addresses in IPset as counts in ranges, concisely:

```
rwsetcat --ip-range list.set
```

List only ranges with 10 or more addresses in set:

```
rwsetcat --ip-range list.set | sed -e '/^ *[1-9]|/d'
```

rwsetcat Parameters

Parameters

Uses the same format parameters as rwcut; does not support plugins, pmap, or Python.

MIN-NAME	DESCRIPTION	ARGUMENTS
cidr	Print IPs in CIDR block notation	zero or one (opt.)
count	Print the number of IPs	none
help	Print usage summary	none
ip-range	Print IPs as ranges of count—low—high	none
net	Summarize CIDR blocks in set Proto v4: (uses asv4 policy, v6: (uses force policy), or blank (counts as v4:) Disp is a comma-separated list of CIDR lengths, letters, or blank (counts as H) For v4, letters are T=0,A=8,B=16,C=24, X=27,orH=32 For v6, letters are T=0,H=128 Summ is blank (for no summary) or a combination of S—use later list or defaults for v4, A, B, C, X, for v6, 48, 64 /—(if no later list) previous Disp as summary blocks /list—specify another list in Disp format as summary blocks	ProtoDispSumm (opt.)
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
print-file	Print filename as they are processed	zero or one (opt.)
print-ips	Also print IPs if count or statistics parameter present	none
print-stat	Print set statistics	none
version	Print program version	none

rwsetmember

Determine whether IP addresses are members of IP sets

Syntax Summary

`rwsetmember` *`pattern parameters filename ...`*
Parameters and set files are optional; pattern is required

Examples

Test if 172.18.14.3 is in set rfc1918.set:

```
rwsetmember 172.18.14.3 rfc1918.set
```

List which files ending in '.set' contain 172.18.14.3:

```
rwsetmember 172.18.14.3 *.set
```

Categorize address in temp.txt based on presence in rfc1918.set:

```
for addr in $(cat test.txt); do
  rwsetmember --quiet $addr rfc1918.set; r1918=$?
  if [[ $r1918 == 0 ]]; then
    echo $addr " private"
  else
    echo $addr " public"
  fi
done
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
count	print number of address match pattern per file	none
help	Print command summary	none
quiet	Suppress listing file name on match	none
version	Print this program's version	none

rwsort

Sort binary flow records, merging files if required.

Syntax Summary

`rwsort --fields=key-fields` *`parameters filename ...`*
Parameters and flow files are optional; `--fields` is required

Examples

Ordering flow file by start time:

```
rwsort --fields=stime flows.rw >sorted.rw
```

Ordering flow file by source IP address, then by time:

```
rwsort --fields=sip,stime --output=src-time.rw flows.rw
```

Merging two flow files and ordering by start time:

```
rwsort --fields=stime one.rw two.rw >time-order.rw
```

Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
compress	Select compression	comp. opt.
help	Print command summary	none
help-fields	Print field descriptions	none
note-add	Put *arg* in file header	string
note-file	Put contents of *arg* in file header	filename
output	Output destination	filename
plugin	Load given plugin(s) to add fields	filename
pmap-file	Prefix map file to read. Use before `--fields`	map:filename
presort	Merge only (do not sort)	none
print-file	Print names of input files as they are opened	none
python	Use Python code to extend processing	filename
reverse	Reverse the sort order	none
site-conf	Site configuration file	filename
sort-buff	Memory allocation for sort buffer	integer[`k,M,G`]
temp	Store temporary files here	dirname
version	Print this program's version	none
xarg	Read input file names from file or pipe	filename (opt.)

rwcount

Summarize binary flow records across time.

Syntax Summary

`rwcount` *`parameters filename ...`*

All parameters are optional.

Examples

Generating 30-second counts (the default) of records from standard input, with data proportional to time:

```
rwcount >30-sec.txt
```

Generate five-minute counts from file, with data proportional to time:

```
rwcount --bin-size=300 flows.rw >five-min.txt
```

Generate hourly counts in comma-separated variable (CSV) format, with data only in start time block, from file (including `sed` command to add space after comma):

```
rwcount --bin-size=3600 --delim=',' --load-scheme=1 flows.rw \
| sed -e 's/,/, /g' >hr.csv
```

Generate 12-minute counts (calculated in-line), with data proportional to time:

```
rwcount --bin-size=$((12*60)) flows.rw >12-min.txt
```

Common bin-size Values

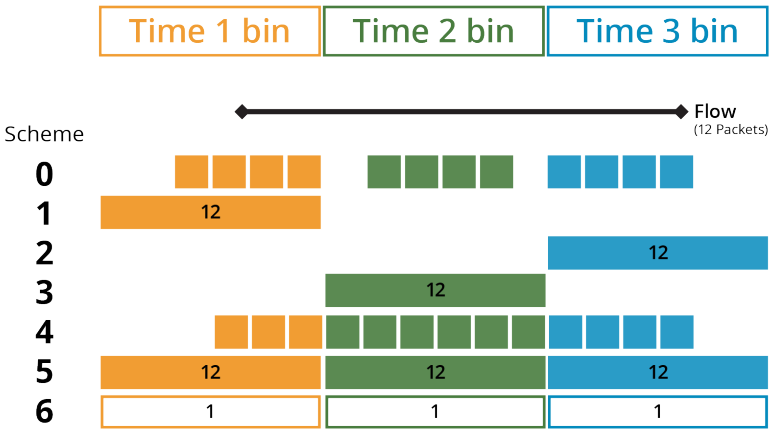
INTERVAL	BIN-SIZE VALUE
5 min	300
10 min	600
15 min	900
30 min	1800
Hour	3600
Day	86400
Week	604800

`rwcount` Parameters

Also format parameters as `rwcut` takes, but no plugin, pmap, or python

MIN-NAME	DESCRIPTION	ARGUMENTS
bin-size	Size of bins in seconds (default 30.000)	decimal
bin-slots	Print bin labels using the internal bin index	none
copy	Copy all input SiLK flows to given pipe or file	filename
end	Print bins until this time	time
epoch	Print bin labels using epoch time	none
help	Print command summary	none
load	Specifies handling of flows that span bins 0—split volume EVENLY across the bins 1—fill FIRST appropriate bin with complete volume 2—fill LAST appropriate bin with complete volume 3—fill CENTERMOST bin with complete volume 4—split volume into bins proportional to time ACTIVE 5—assign MAXIMUM possible volume for each bin 6—assign MINIMUM possible volume for each bin *(for 5 and 6, the sum of all bin values may not match the total volume)*	integer

Load Schemes



`rwcount` **Parameters** *(continued)*

MIN-NAME	DESCRIPTION	ARGUMENTS
output	Send output to given file path (default `stdout`)	filename
pager	Program to invoke to process output	filename
print-file	Print file names as they are opened	none
site-conf	Specify location of the site configuration file	filename
skip-zero	Don't print bins that have no flows	none
start	Print bins from this time forward	time
version	Print this program's version	none
xarg	Read input file names from pipe or file	filename (opt.)

Formatted Output

```
$ rwcount --bin-size=600 < flows.rw
                Date|      Records|          Bytes|       Packets|
2015/06/17T14:00:00|       466.00|       798757.00|       3423.00|
2015/06/17T14:10:00|       394.00|       104668.00|       1622.00|
2015/06/17T14:20:00|       382.43|       104159.18|       1621.86|
2015/06/17T14:30:00|       393.57|       107100.82|       1670.14|
2015/06/17T14:40:00|      9335.01|     15559931.61|     191709.67|
2015/06/17T14:50:00|     10885.11|     16541697.17|     187619.55|
2015/06/17T15:00:00|         7.70|        75830.56|        897.45|
2015/06/17T15:10:00|         0.17|        21466.66|        383.33|
```

Time stamp for earliest moment in BIN

Number of flow records in BIN

Number of bytes in BIN

Number of packets in BIN

rwstats

Generate top N, bottom N, or descriptive statistics from file.

Syntax Summary

(Two alternative forms)

Generate descriptive statistics:

`rwstats --overall` *`filename ...`*

`rwstats --detail=`*`protocol-list filename ...`*

Overall or by protocol, specify filename or pipe from `stdin`

Generate top/bottom N lists:

`rwstats --fields=`*`fieldlist`* `--values=`*`vallist`*
`--top` *`bound options filename ...`*

`rwstats --fields=`*`fieldlist`* `--values=`*`vallist`*
`--bottom` *`bound options filename ...`*

where vallist is a comma-separated list of bytes, packets, flow records, sip-distinct, dip-distinct or distinct:KEYFIELD

Examples

Find 10 highest-volume IP pairs:

```
rwstats --fields=sip,dip --values=bytes --top --count=10 flows.rw
```

Find all destination ports that get more than ten percent of the traffic by frequency:

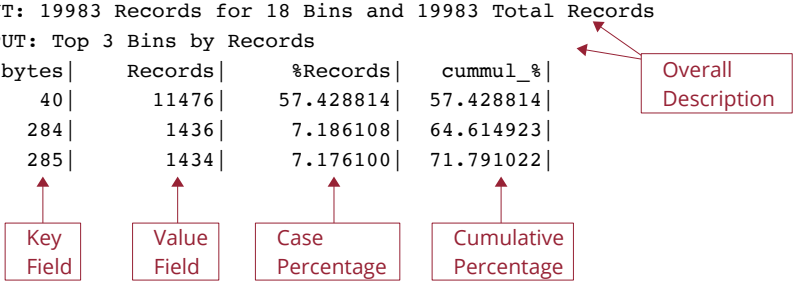
```
rwstats --field=dport --values=records --top --percent=10 flows.rw
```

Print descriptive statistics on traffic volumes:

```
rwstats --overall flows.rw
```

Formatted Output

```
$ rwstats --fields=bytes --values=records --top --count=3 maybe.rw
INPUT: 19983 Records for 18 Bins and 19983 Total Records
OUTPUT: Top 3 Bins by Records
    bytes|     Records|      %Records|   cummul_%|
       40|       11476|    57.428814|  57.428814|
      284|        1436|     7.186108|  64.614923|
      285|        1434|     7.176100|  71.791022|
```



`rwstats` Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Bounds	
bottom	Apply bounds from within key with lowest value	none
count	Specify N (0 means print all)	integer
percent	Specify percent for bound. Only for bytes, packets, or flows	decimal
threshold	Specify value for bound (not from plugins)	integer
top	Apply bounds from key with highest value	none
	Options	
bin-time	Specify bin size for time keys	decimal
copy	Copy all input SiLK flows to given pipe or file	filename
help	Print command summary	none
help-fields	Print field descriptions	none
ipv6-policy	Specify how to handle IPv4/v6	(see `rwcut` descr.)
legacy-help	Print help for legacy switches	none
no-per	Don't print the percentage columns	none
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
presort	Assume input has been presorted for fields	none
print-file	Print names of input files	none
site-conf	Specify location of the site configuration file	filename
temp	Store temporary files in this directory	dirname
version	Print program version	none
xarg	Read input file names form file or pipe	filename (opt.)

rwuniq

Summarize traffic volumes based on unique combinations of flow record fields.

Syntax Summary

`rwuniq --fields=`*`fieldlist`* `--values=`*`vallist options filename ...`*
Options and filename are optional; values may be replaced by counting parameters

Examples

Generate byte count totals of protocols grouped by hour from a file:

```
rwuniq --fields=proto,stime --bin-time=3600 --values=bytes flows.rw
```

Generate byte count totals and number of source addresses of high-volume flows by destination ports from a file:

```
rwuniq --fields=dport --values=bytes,distinct:sip --bytes=10000- maybe.rw
```

Generate contrasting views of traffic by size and by source port:

```
rwuniq --fields=bytes --values=records,distinct:dip \
--output=bytes --copy=stdout flows.rw \
| rwuniq --fields=sport --values=records,distinct:dip --output=sport.txt
```

Count source ports per source address:

```
rwuniq --fields=sip --values=distinct:sport flows.rw
```

Count bytes, packets, and flow records by protocol, reporting in protocol number:

```
rwuniq --fields=protocol --values=bytes,packets,flows --sort flows.rw
```

Generate byte and flow counts per source address, limiting output to bins between 5,000 and 10,000 bytes:

```
rwuniq --fields=sip --values=bytes,records --threshold=bytes=5000-10000 \
--sort flows.rw
```

Formatted Output

```
$ rwuniq --fields=dport --values=bytes,distinct:sip maybe.rw
   dPort|          Bytes|  sIP-Distin|
      22|        2492768|           1|
    7051|         636478|           1|
    7052|         635862|           1|
```

`rwuniq` Parameters

MIN-NAME	DESCRIPTION	ARGUMENTS
	Option Parameters	
bin-time	Specify bin size for time keys	decimal
copy	Copy all input SiLK flows to given pipe or file	filename
epoch-time	Prints times as count of seconds since epoch	none
fields	Field combination for bins	fieldlist
help	Print command summary	none
help-fields	Print field descriptions	none
ipv6-policy	Specify how to handle IPv4/v6	(see `rwcut` page)
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
presort	Assume input has been presorted for fields	none
print-file	Print names of input files	none
site-conf	Specify location of the site configuration file	filename
temp	Store temporary files in this directory	dirname
version	Print program version	none
xarg	Read input file names from file or pipe	filename (opt.)
	Counting Parameters	
all	Bytes, packets, flows, stime, and etime	none
bytes	Sum bytes in each binj	int-range
dip-dist	Count distinct destination addresses in each bin	int-range
flows	Count flow records in each bin	int-range
packets	Sum packets in each bin	int-range
sip-dist	Count distinct source addresses in each bin	int-range
threshold	Limit output using value limits; argument is `value=MIN` or `value=MIN-MAX`	
values	Value(s) to compute: `bytes`, `packets`, `records`, `distinct:`*`KEYFIELD`*, `stime-earliest`, `flows`, `etime-latest`	valuelist

Notes

IP Protocols

NUM	NAME	DESCRIPTION	HEADER BYTES	
			IPV4	IPV6
0	HOPOPT	IPv6 Hop-by-Hop		48
1	ICMP	Internet Control Messages	28	
2	IGMP	Internet Group Management	28	
3	GGP	Gateway-to-Gateway	24	
4	IPv4	IPv4 Encapsulation	40	60
6	TCP	Transmission Control	40	60
8	EGP	Exterior Gateway	30	50
9	IGRP	Interior Gateway	28	
17	UDP	User Datagram	28	48
27	RDP	Reliable Data	38	58
28	ITRP	Internet Reliable Transaction	28	48
41	IPv6	IPv6 Encapsulation	60	80
43	IPv6-Route	IPv6 Routing Header		48
44	IPv6-Frag	IPv6 Fragment Header		48
46	RSVP	Reservation Protocol	28	48
47	GRE	Generic Route Encapsulation	28	44
50	ESP	Encap Security Payload	28	48
51	AH	Authentication Header	32	52
53	SWIPE	IP with Encryption	28	48
58	ICMP	ICMP for IPv6		44
59	NoNxt	No Next Header for IPv6		48
60	IPv6-Opts	Destination Options for IPv6		48
88	EIGRP	Enhanced Interior Gateway Routing	40	60
98	ENCAP	Encapsulation Header	28	48
99		Private Encryption	20	40
132	SCTP	Stream Control Transmission	32	52
143-252		*Unassigned*	--	
253-254		*Experimental*	--	
255		*Reserved*	--	

SiLK Commands *(continued)*

Text Output SiLK Tools

PAGE	TOOL SUMMARY
	rwbagcat—display and characterize bag content
	rwaggbagcat—display and characterize aggregate bag content
	rwcompare—determine if two flow files are identical
23	rwcount—time-series counts
8	rwcut—text from flows
	rwfglob—list repository files from rwfilter selection parameters
11	rwfileinfo—describe file contents
	rwpcut—display packet fields of PCAP data
	rwpmapcat—display pmap content
	rwpmaplookup—display pmap label for IP address
	rwresolve—perform DNS lookup from IP address text
	rwscan—apply scan detection models to flows
	rwscanquery—query the network scan database
19	rwsetcat—display IP set content
21	rwsetmember—determine which IP sets have this address
12	rwsiteinfo—display repository information as configured
26	rwstats—generate top N/bottom N counts or protocol statistics
28	rwuniq—generate aggregate counts

(Tools without page numbers are not in this guide.)

(See front cover for binary output tools.)

For More Information

— https://tools.netsa.cert.org/silk/docs.html

— *Network Traffic Analysis With SiLK*—tutorial on how to use the SiLK tools to analyze network traffic

— *PySiLK; SiLK in Python*—reference guide for manipulating SiLK flow data within Python

— *The SiLK Reference Guide*—every SiLK manual page in a single document

— *SiLK Installation Handbook*—instructions on configuring, building, and installing SiLK at your site

Copyright 2019 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

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.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

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. DM19-1187