Beginning Analysis with SiLK

Geoffrey Sanders

Nathaniel Richmond

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Homeland Security 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 sponsored by the United States Department of Defense.

DM18-0043

Learning Objectives

At the end of this class, analysts will have the knowledge and skills needed to perform the following tasks:

- Describe SiLK and network flow.
- Display SiLK site configuration using the `rwsiteinfo` command.
- Retrieve network flow records using the `rwfilter` command.
- Manipulate network flow records using the `rwcut` and `rwsort` commands.
- Count and profile network flow records using the `rwstats` and `rwuniq` commands.
- Manipulate IPsets using the `rwsetbuild`, `rwsetcat`, `rwsettool`, and `rwset` commands.

Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

What is SiLK?

SiLK - “System for internet Level Knowledge”

Collection of network flow traffic analysis tools from the CERT Situational Awareness group at the Software Engineering Institute (SEI), Carnegie Mellon University (CMU)

What SiLK Enables

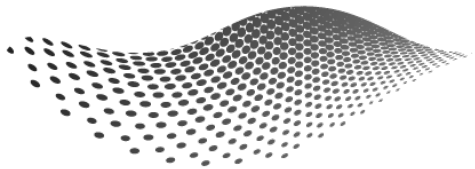
Security analysis of large networks

Efficient collection, storage, and analysis of network flow data

Rapid query of large historical traffic data sets

Ideally suited for distributed enterprise networks

Suitable for small (home) networks as well



SiLK

Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

What is Network Flow?

A metadata record of related packets

- log of network activity (not packet capture)
- similar to a phone bill (call detail record)

Content of messages is *not* recorded

- highly compact
  - increased retention
  - reduced processing
- privacy support
- reduced encryption impact

IP Flow Information Export (IPFIX) common standard

‘Flow’, ‘flow record’, and ‘record’ are commonly used in place of ‘network flow’

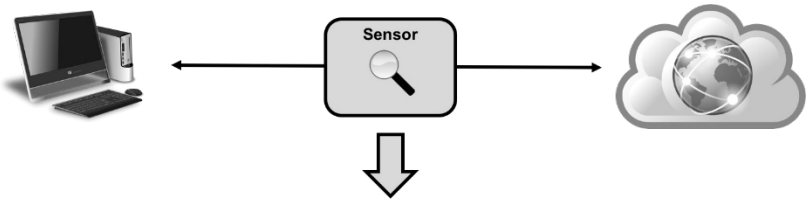
Call Detail Records

LATITUDE	LONGITUDE	DATE	TIME	NUMBER	NAME	DURATION
44.50880 N	73.18223 W	1/28/2008	0917	802-555-1234	Chittenden Bank	0:10:17
44.50880 N	73.18223 W	1/28/2008	0942	802-555-8673	Poopsie LaRue	0:01:03
44.50880 N	73.18223 W	1/28/2008	0945	802-555-9201	Hanley Strappman	0:05:32
44.27834 N	73.21263 W	1/29/2008	2205	802-555-7758	Verizon Voice Mail	0:01.13
44.27834 N	73.21263 W	1/29/2008	1532	802-555-4492	Widgets LLC	0:03:47
44.27834 N	73.21263 W	1/29/2008	2209	802-555-7758	Verizon Voice Mail	0:00.36
44.50880 N	73.18223 W	1/30/2008	0830	202-555-1818	British Embassy	0:18:12
44.27834 N	73.21263 W	1/30/2008	2208	802-555-7758	Verizon Voice Mail	0:00.53
44.27834 N	73.21263 W	1/30/2008	2211	802-555-8673	Poopsie LaRue	0:06:18
44.50880 N	73.18223 W	1/31/2008	0903	202-555-1843	British Embassy	0:03:21
44.50880 N	73.18223 W	1/31/2008	0908	416-555-9834	British Embassy	0:22:04
44.4143 N	73.03561 W	1/31/2008	1047	802-555-9201	Hanley Strappman	0:01:02
44.4143 N	73.03561 W	1/31/2008	1050	213-555-2761	M. Fendell	0:09:06
44.25295 N	72.58229 W	1/31/2008	1127	802-555-9201	Hanley Strappman	0:05:38

What's in a Network Flow Record?

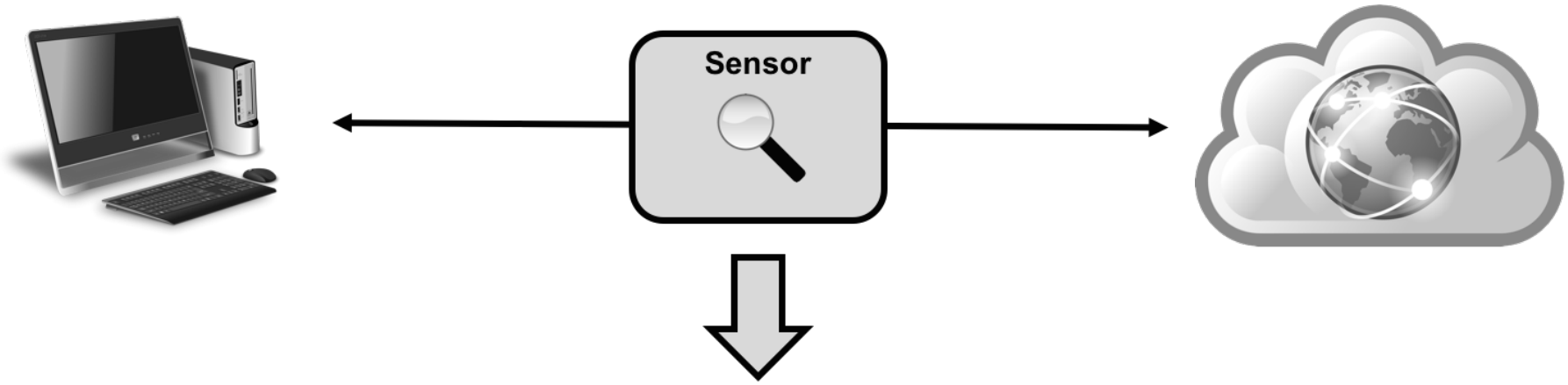
Fields found in network flow records

- source address, destination address
- source port, destination port, Internet Control Message Protocol [ICMP] type and code
- IP [transport] protocol
- bytes, packets
- TCP flags
- start time, end time, duration
- sensor identity
- flow termination conditions
- application-layer protocol



Network Flow Record					
Source Address	Source Port	Destination Address	Destination Port	Start Time	End Time
Bytes	Packets	TCP Flags	Application Label	Protocol	Duration
Sensor Name	Source Country Code	Destination Country Code	Sensor Type	Attributes	Custom Fields

Network Flow Example



Network Flow Record					
Source Address	Source Port	Destination Address	Destination Port	Start Time	End Time
Bytes	Packets	TCP Flags	Application Label	Protocol	Duration
Sensor Name	Source Country Code	Destination Country Code	Sensor Type	Attributes	Custom Fields

How are Flows Different from Packets?

	Flow	Packet
Unit	Record	Packet (IP)
Composition	Tuple	Header and Payload
Contents	Metadata Summary	Data Unit
Size	Small (52 or 88 bytes)	Variable (20+ bytes)
Trade-Offs	Data payload, retention, and query response time	

Which Packets Combine into a Flow?

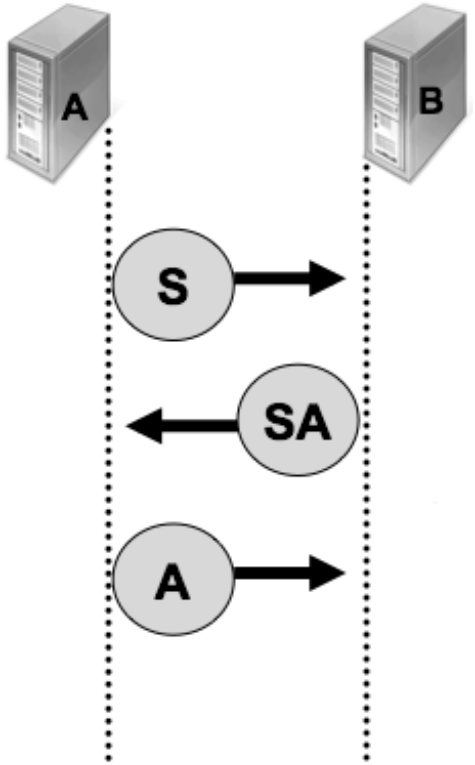
A flow is an aggregated record of packets

SiLK flows are identified by five attributes (5-tuple):

- source IP address
- destination IP address
- source port
- destination port
- transport protocol (any of about 130 in use)

SiLK flows are unidirectional:

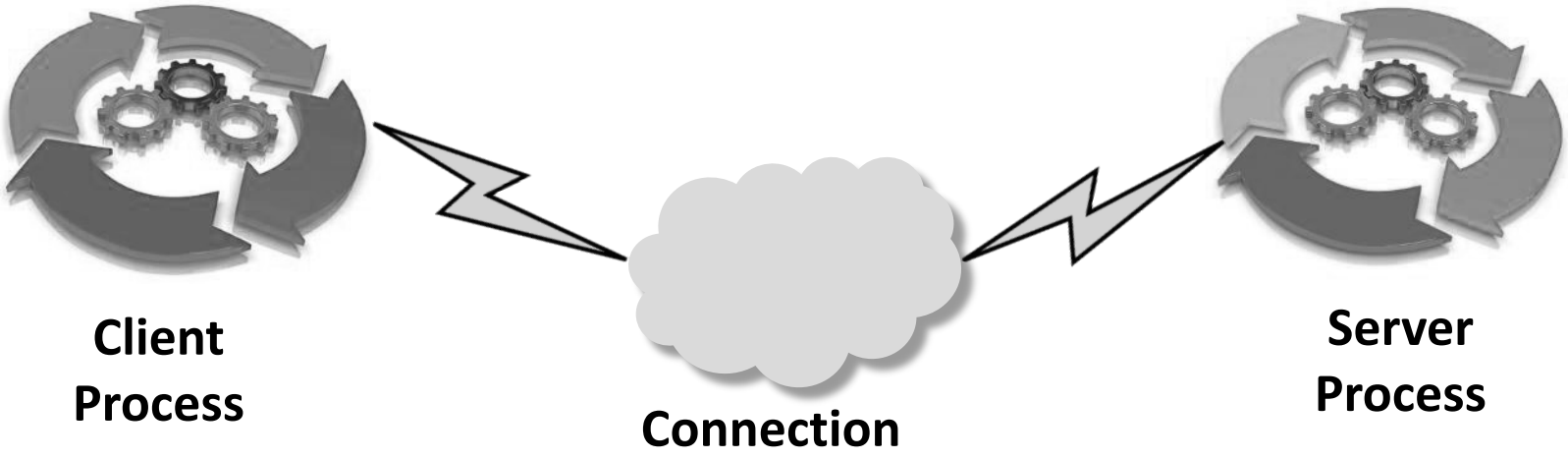
- newly observed attributes, new flow
- previously observed attributes, update flow



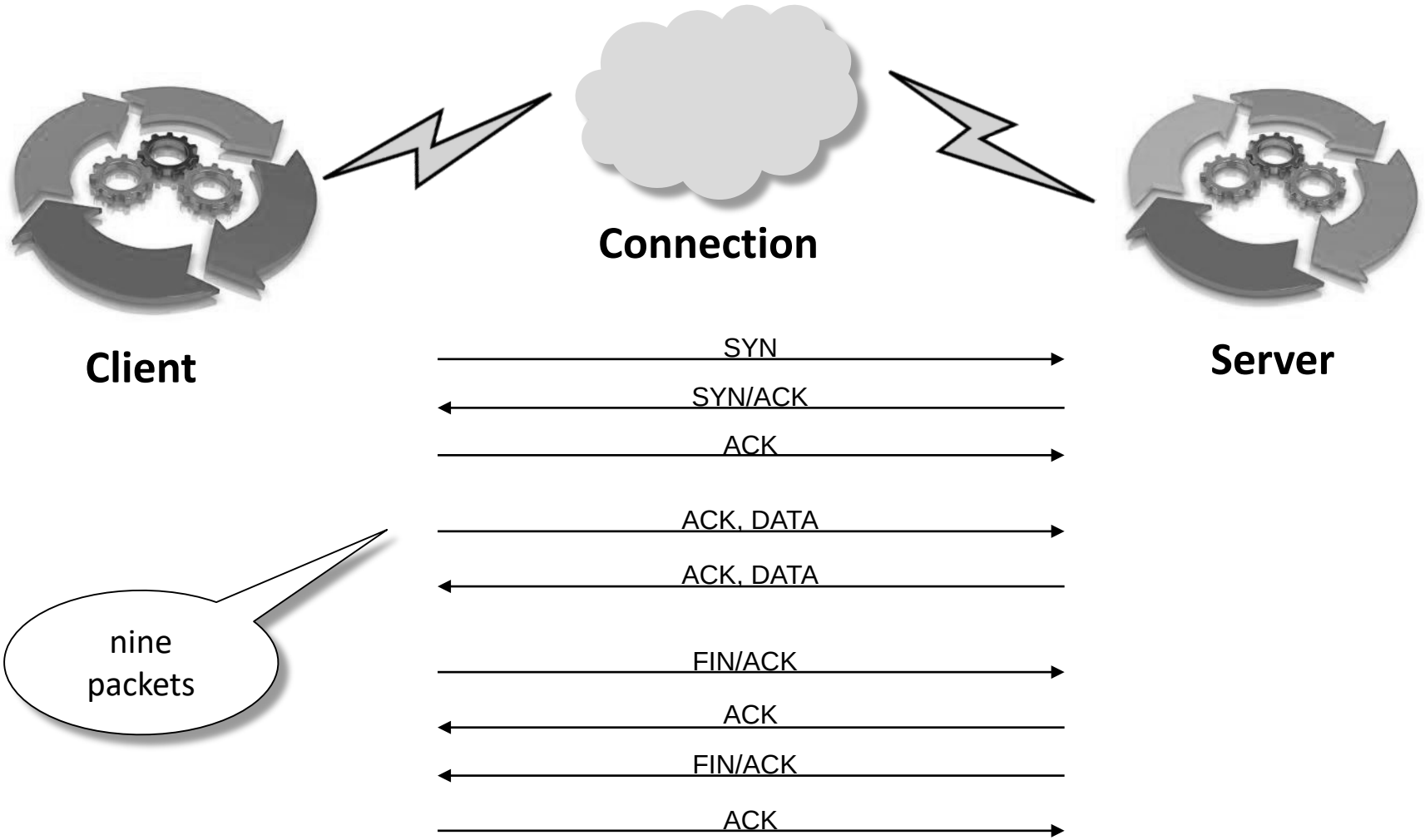
TCP/IP Socket Example

TCP/IP SOCKET
IP address: 10.0.0.1
L4 protocol: TCP
Ephemeral port #

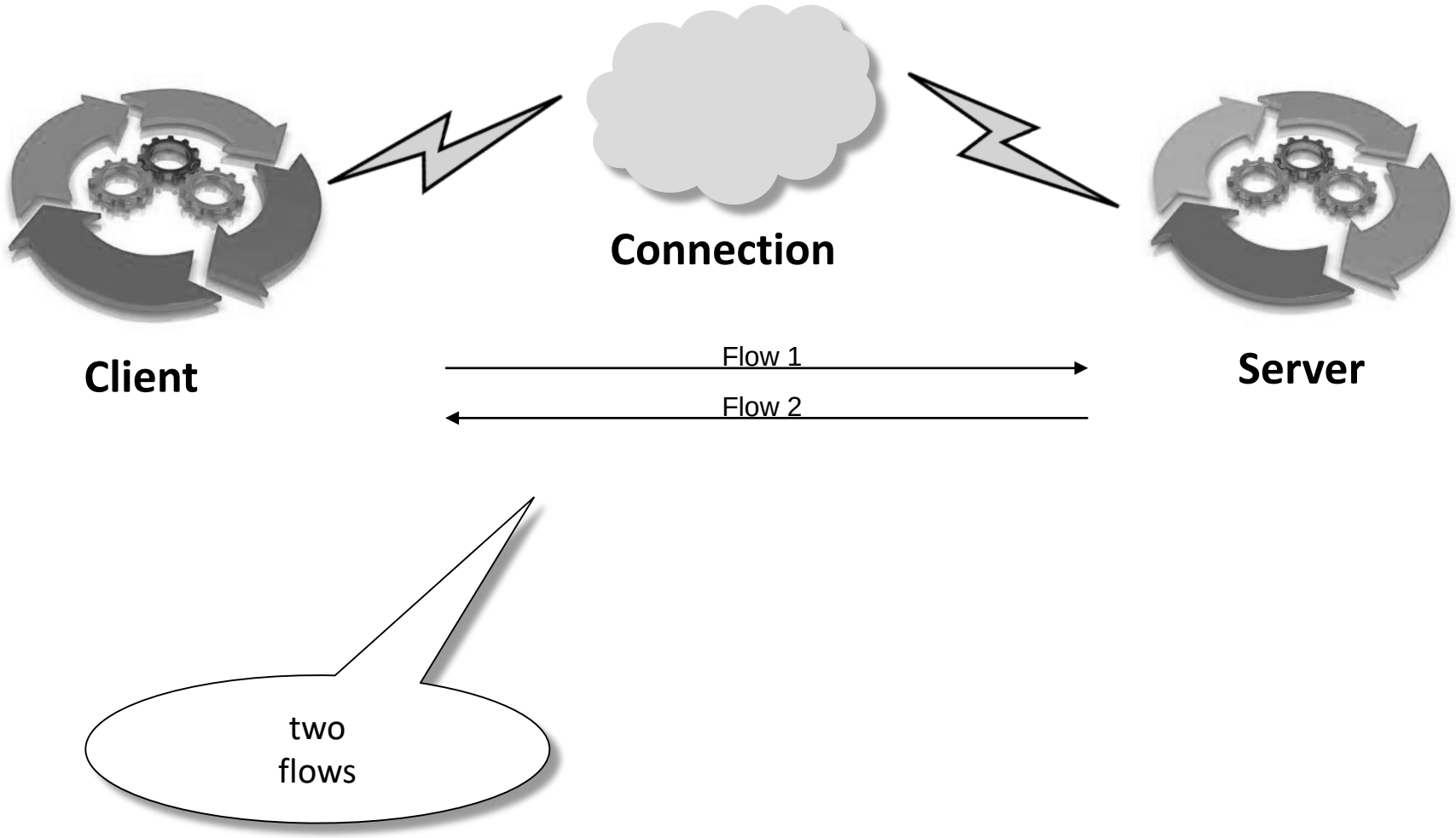
TCP/IP SOCKET
IP address: 203.0.113.1
L4 protocol: TCP
Well-Known Port #



TCP Socket (Packets)



TCP Socket (Flows)



How is Network Flow Used?

Investigation analysis
- past network events
- automated report generation
- forensics (what happened before an incident?)

Descriptive analysis
- profiling/categorizing

Directed analysis (hunt)
- looking for specific malicious behavior

Exploratory analysis
- looking for the unusual

Predictive analysis
- projecting future behavior

Popular Network Flow Analytics

Service inventory

Event precursors

Policy violations

Top-N web servers

Blacklist testing

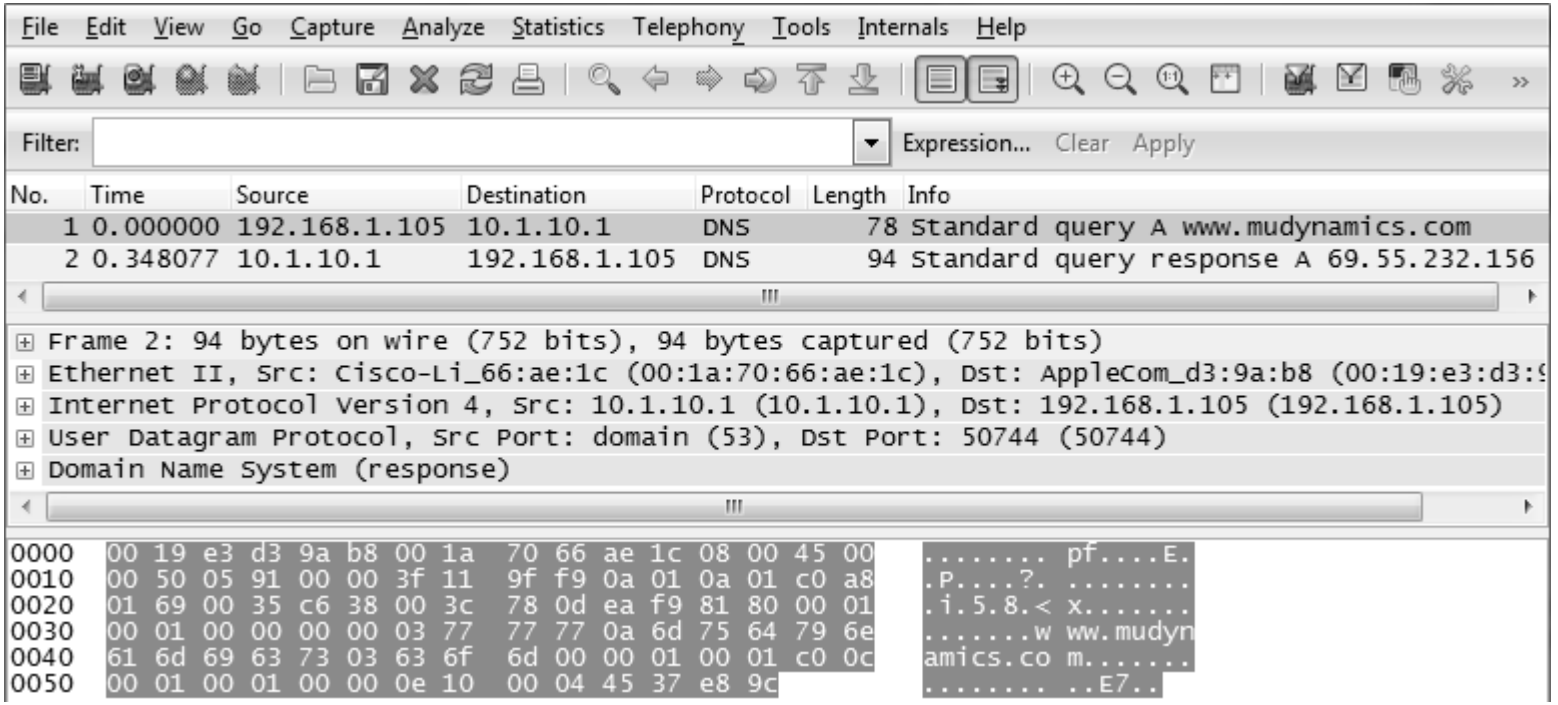
Indicators of Compromise (IoC)

Service interruptions

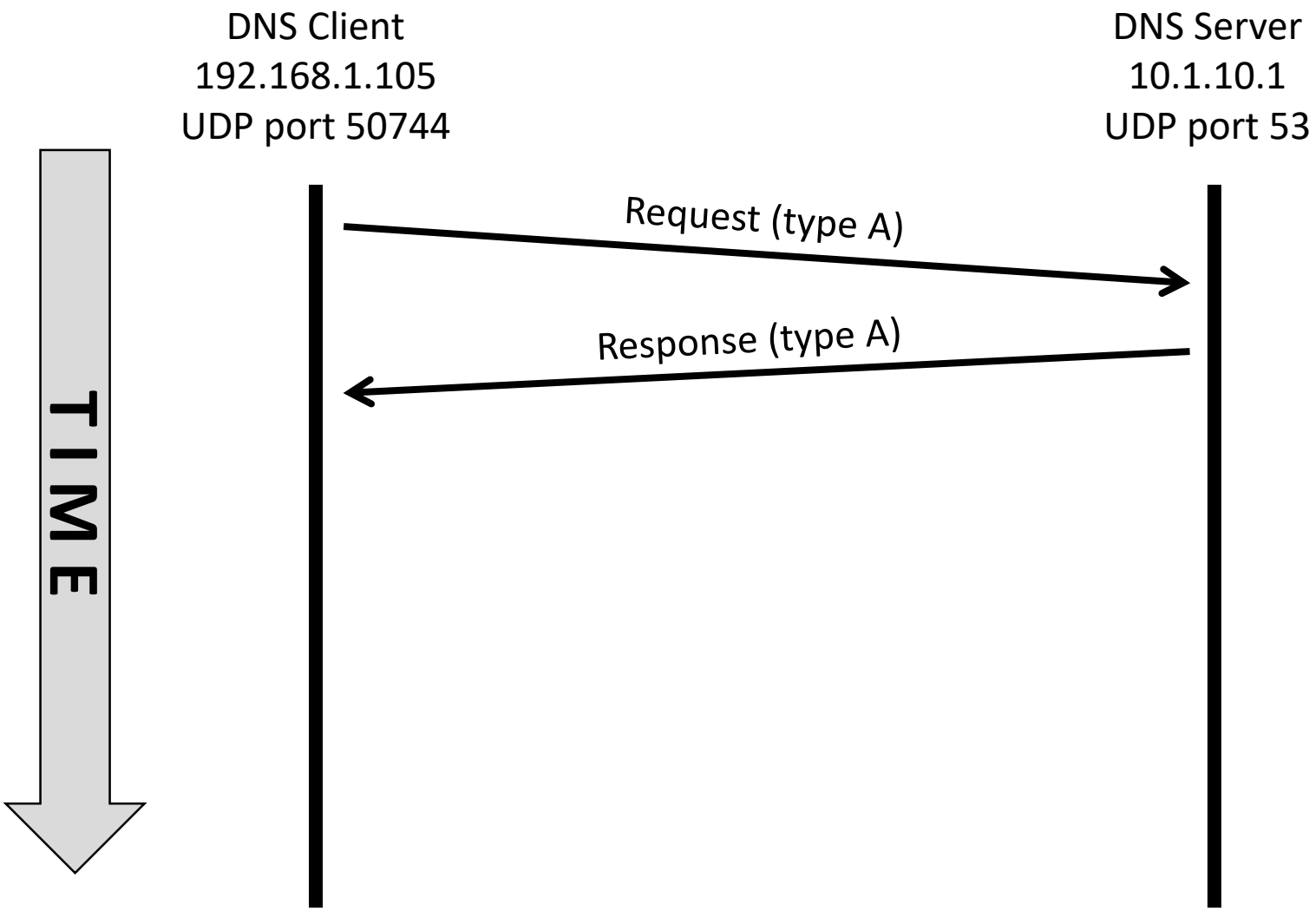
Spam behavior

IP address audits

DNS Packets Viewed in Wireshark



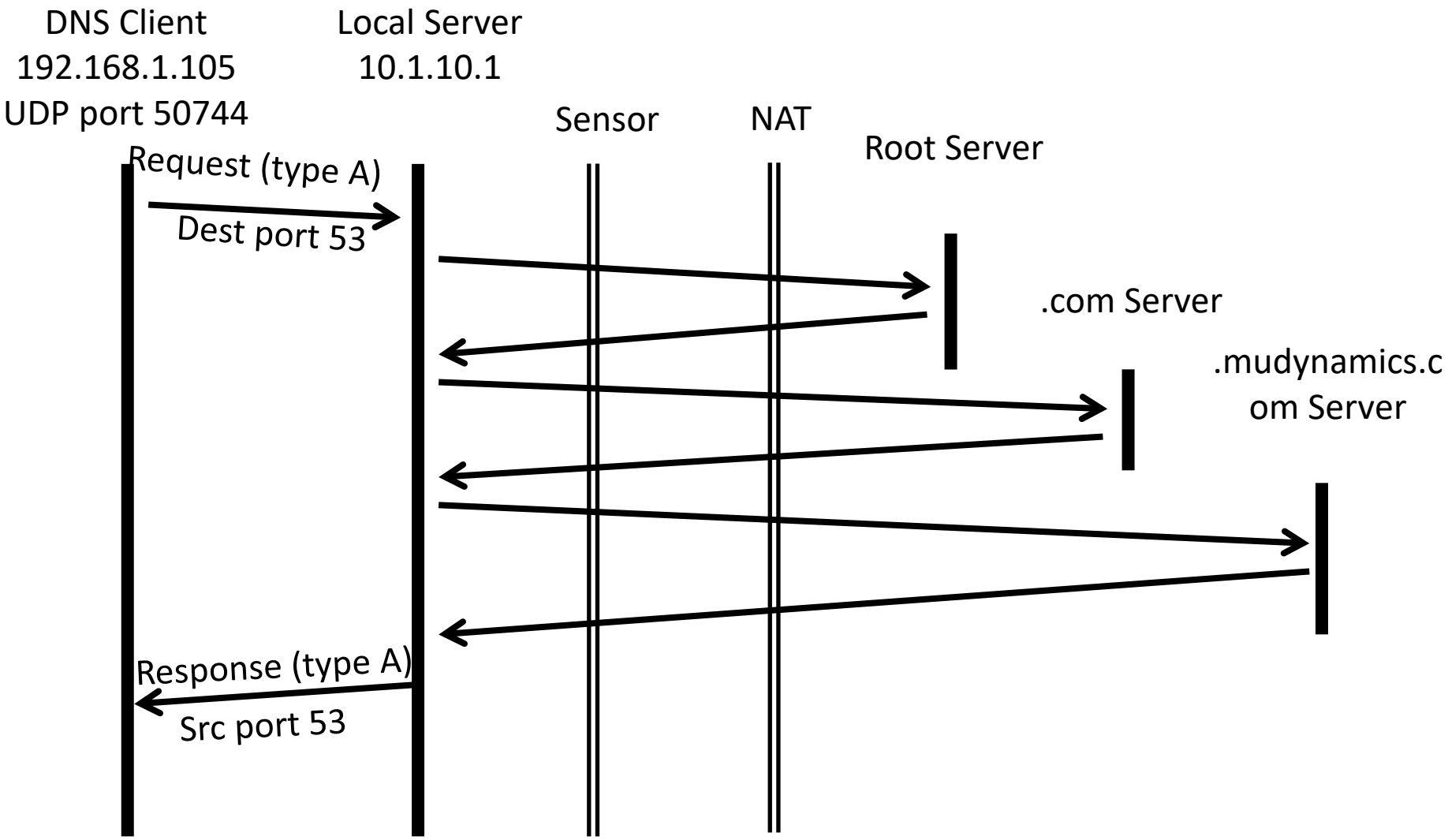
Simple DNS Sequence Diagram



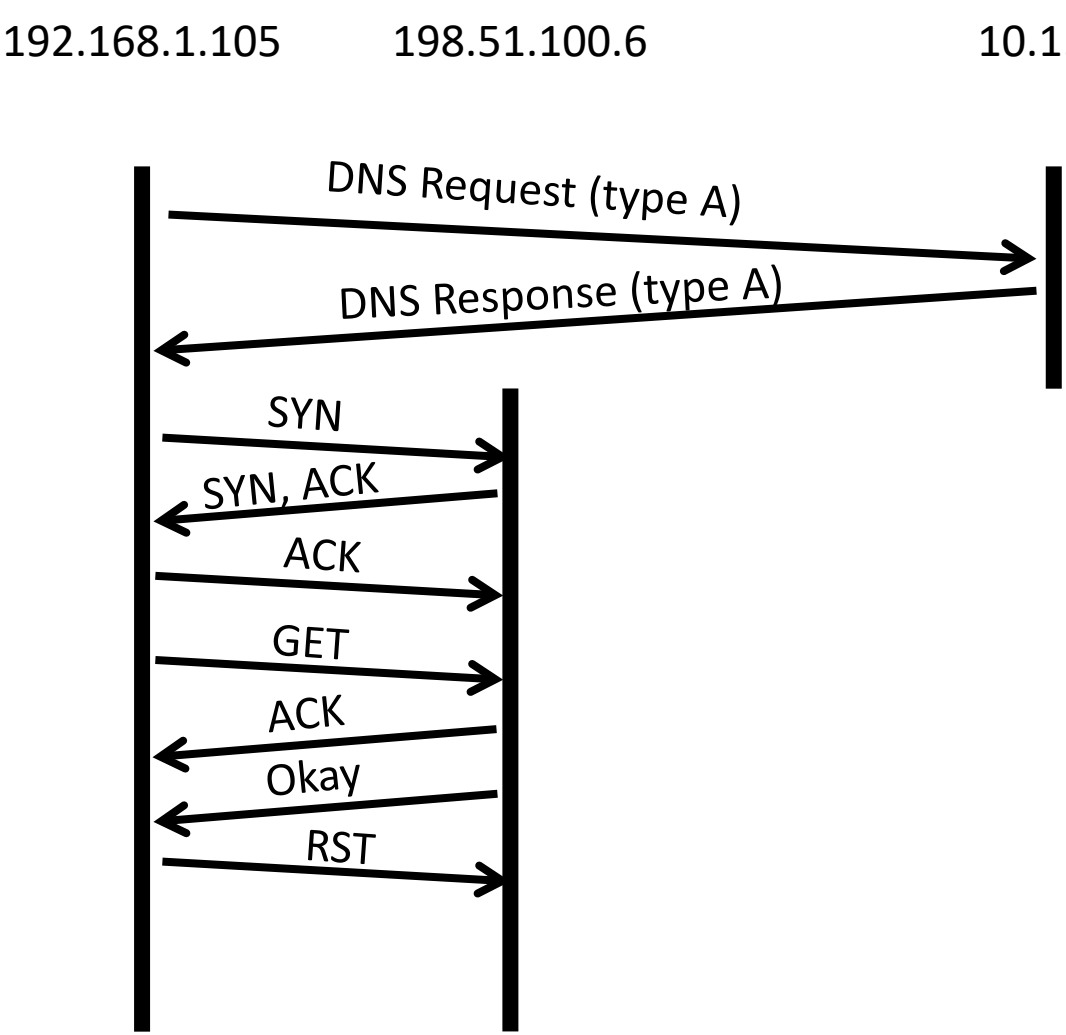
SiLK DNS (`rwcut`) Output

```
          sIP|          dIP|sPort|dPort|pro|packets|bytes|sensor|type|
192.168.1.105|    10.1.10.1|50744|   53| 17|      1|   64|    S1| out|
    10.1.10.1|192.168.1.105|   53|50744| 17|      1|   80|    S1|  in|
```

Operational DNS Sequence Diagram



HTTP Sequence Diagram



What Is This?

```
          sIP|          dIP|sPort|dPort|pro|packets|flags|initF|  type|
192.168.1.105|    10.1.10.1|50744|   53|  17|      4|      |      |   out|
    10.1.10.1|192.168.1.105|   53|50744|  17|      4|      |      |    in|
192.168.1.105| 198.51.100.6|49152|   80|   6|      4| SRPA| S    |outweb|
 198.51.100.6|192.168.1.105|   80|49152|   6|      3| S PA| S  A| inweb|
```

What Is This?

```
          sIP|          dIP|sPort|dPort|pkts| bytes|flags|              sTime|
88.187.13.78|71.55.40.204|40936|   80|  83|  3512|FS PA|2010/12/08T11:00:01|
71.55.40.204|88.187.13.78|   80|40936|  84|104630|FS PA|2010/12/08T11:00:01|
88.187.13.78|71.55.40.204|40938|   80| 120|  4973|FS PA|2010/12/08T11:00:04|
71.55.40.204|88.187.13.78|   80|40938| 123|155795|FS PA|2010/12/08T11:00:05|
88.187.13.78|71.55.40.204|56172|   80|  84|  3553|FS PA|2010/12/08T12:00:02|
71.55.40.204|88.187.13.78|   80|56172|  83|103309|FS PA|2010/12/08T12:00:02|
88.187.13.78|71.55.40.204|56177|   80| 123|  5093|FS PA|2010/12/08T12:00:05|
71.55.40.204|88.187.13.78|   80|56177| 124|157116|FS PA|2010/12/08T12:00:05|
```

What Is This?

```
          sIP|          dIP|sPort|dPort|pro|packets| bytes|flags|initF|
30.22.105.250| 71.55.40.253|52415|   25|  6|     22| 14045|F RPA| S   |
 71.55.40.253|30.22.105.250|   25|52415|  6|     19|  1283|F   P| SA  |
30.22.105.250| 71.55.40.253|52415|   25|  6|      1|    40|     | R   |
```

What Is This?

```
            sIP|             dIP|pro|packets|bytes|                    sTime|
99.217.139.155|  177.252.24.89|  1|      2|  122|2010/12/08T00:04:30.172|
99.217.139.155|177.252.149.249|  1|      2|  122|2010/12/08T00:04:37.302|
99.217.139.155|  177.252.24.52|  1|      2|  122|2010/12/08T00:04:37.312|
99.217.139.155| 177.252.24.127|  1|      2|  122|2010/12/08T00:04:58.363|
99.217.139.155| 177.252.24.196|  1|      2|  122|2010/12/08T00:05:04.327|
99.217.139.155| 177.252.149.30|  1|      2|  122|2010/12/08T00:05:09.242|
99.217.139.155|177.252.149.173|  1|      2|  122|2010/12/08T00:05:12.174|
99.217.139.155|  177.252.24.13|  1|      2|  122|2010/12/08T00:05:14.114|
99.217.139.155|  177.252.24.56|  1|      2|  122|2010/12/08T00:05:15.383|
99.217.139.155| 177.252.24.114|  1|      2|  122|2010/12/08T00:05:18.228|
99.217.139.155| 177.252.202.92|  1|      2|  122|2010/12/08T00:05:22.466|
99.217.139.155| 177.252.202.68|  1|      2|  122|2010/12/08T00:05:23.497|
99.217.139.155| 177.252.24.161|  1|      2|  122|2010/12/08T00:05:30.256|
99.217.139.155|177.252.202.238|  1|      2|  122|2010/12/08T00:05:33.088|
```

Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

Network Flow Metering

Network flow analysis begins with flow generation (‘metering’)

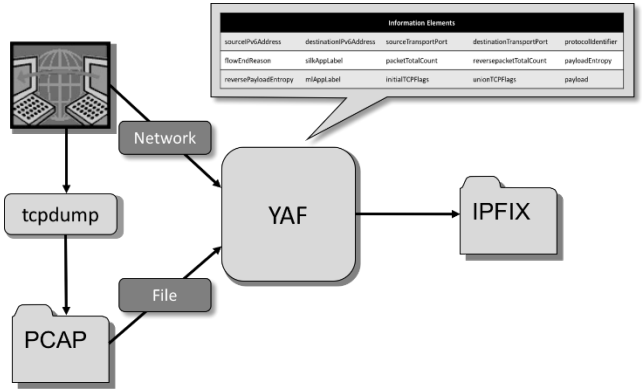
Flowmeters (also called ‘probes’ and ‘sensors’) generate network flow data for SiLK

SiLK reads different network flow meter data standards

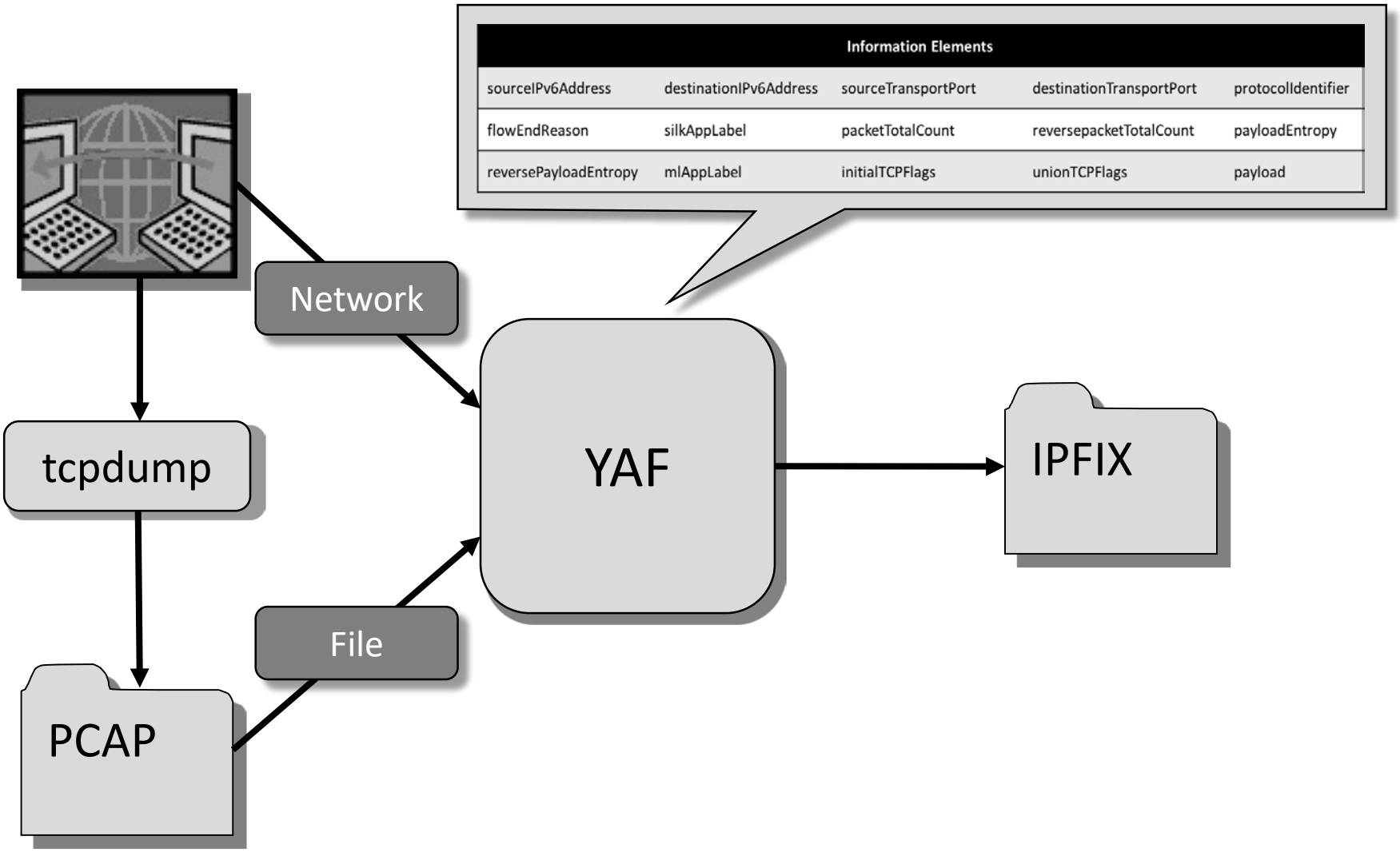
- IPFIX
- NetFlow version 5/9 protocol data units (PDU)

Example network flow meters

- Yet Another Flowmeter (YAF) (IPFIX)
- Routers (NetFlow v5/9)



YAF (Yet Another Flowmeter)



YAF Timing

The YAF flow table (buffer) requires management

Uses ‘timeout’ feature

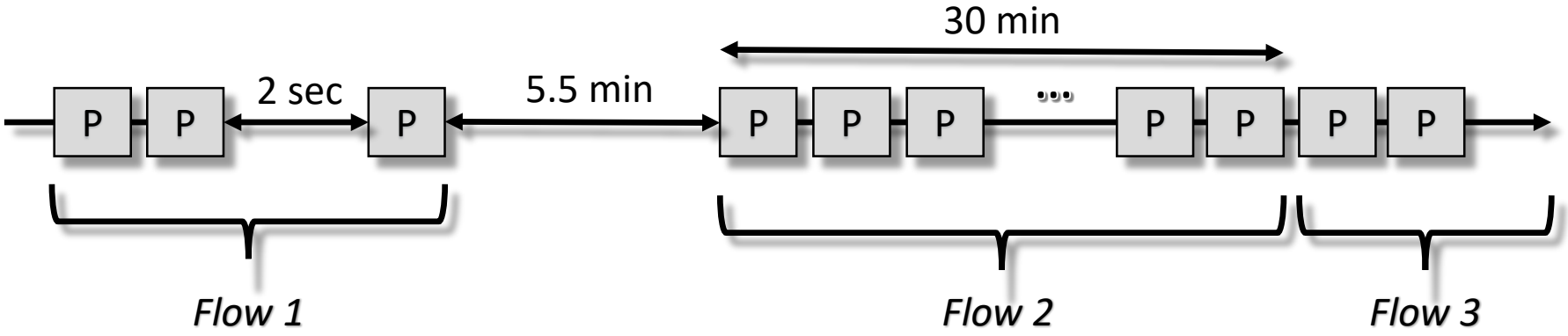
- flush (output) network flow records for collection

Idle timeout

- flush flows without activity at 5 minutes (configurable)

Active timeout

- flush all flows every 30 minutes (configurable)



SiLK Components

SiLK is comprised of a *packing system* and *analysis suite*

Packing system

- collect
- convert
- store

Analysis suite

- read
- partition
- sort
- count
- and display network flow records

Packing

Packing collects, converts, and stores network flow data

Consists of *packer*, *plugin*, and *configuration*

Packer

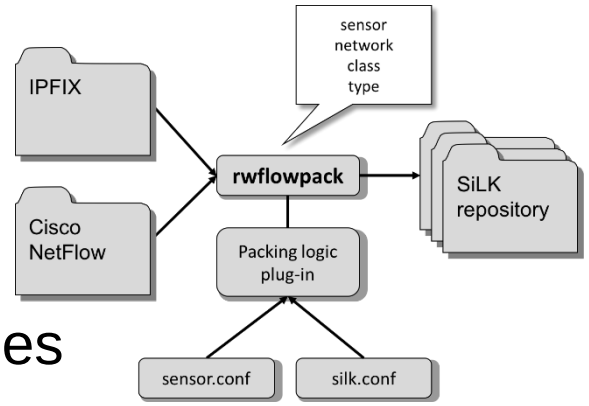
- collect flow data and store in SiLK flow files
- categorize flow records into one or more class and type pairs

Plugin

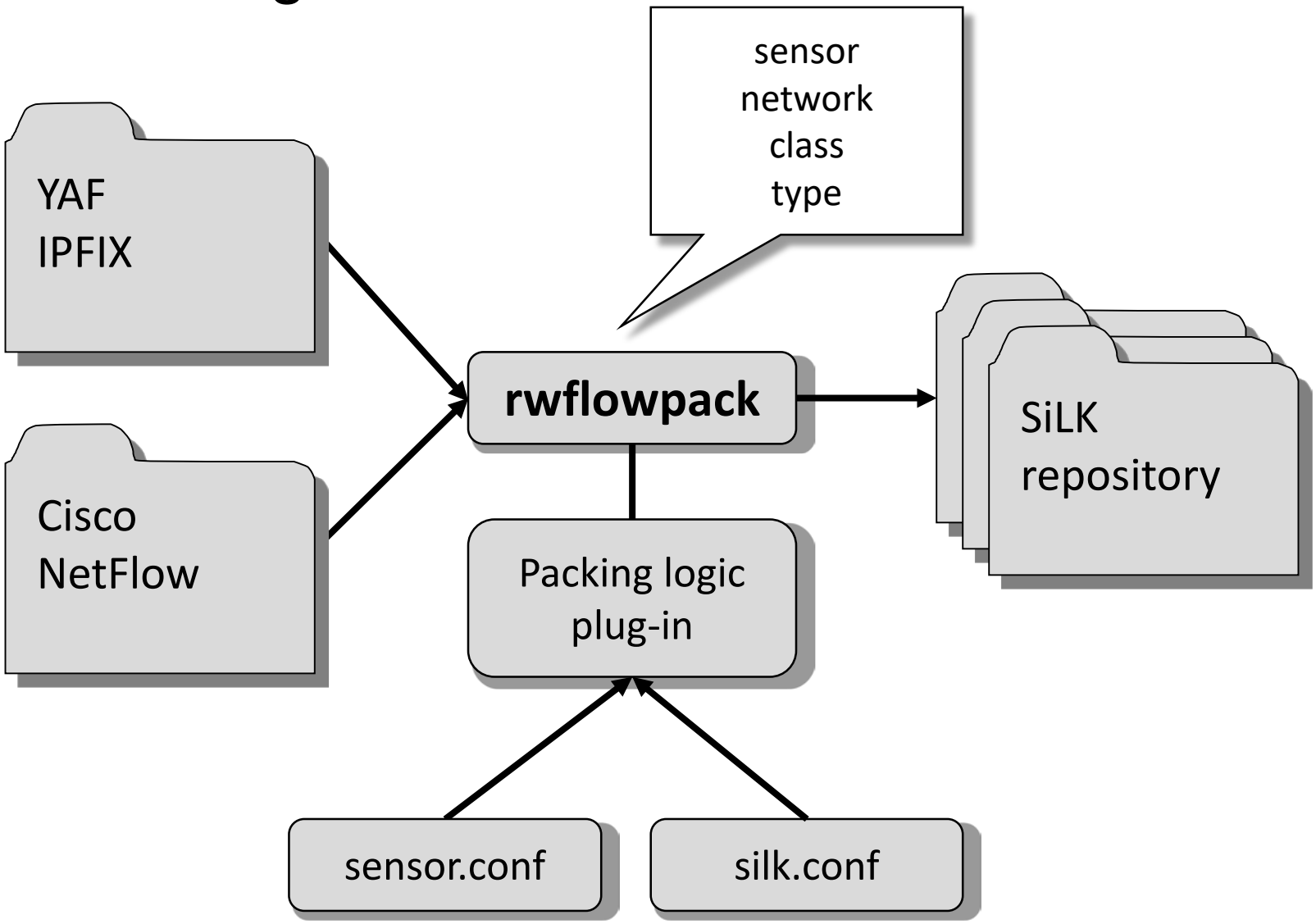
- packing logic

Configuration

- silk.conf - classes, types, and sensors
- sensor.conf - networks, sensors and probes



SiLK Packing



Sensors

Sensors are logical flow collection points

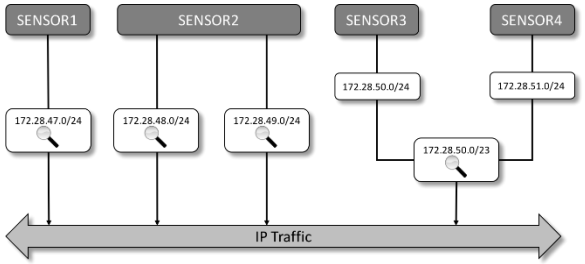
Comprised of

- one or more meters (probes)
- unique names

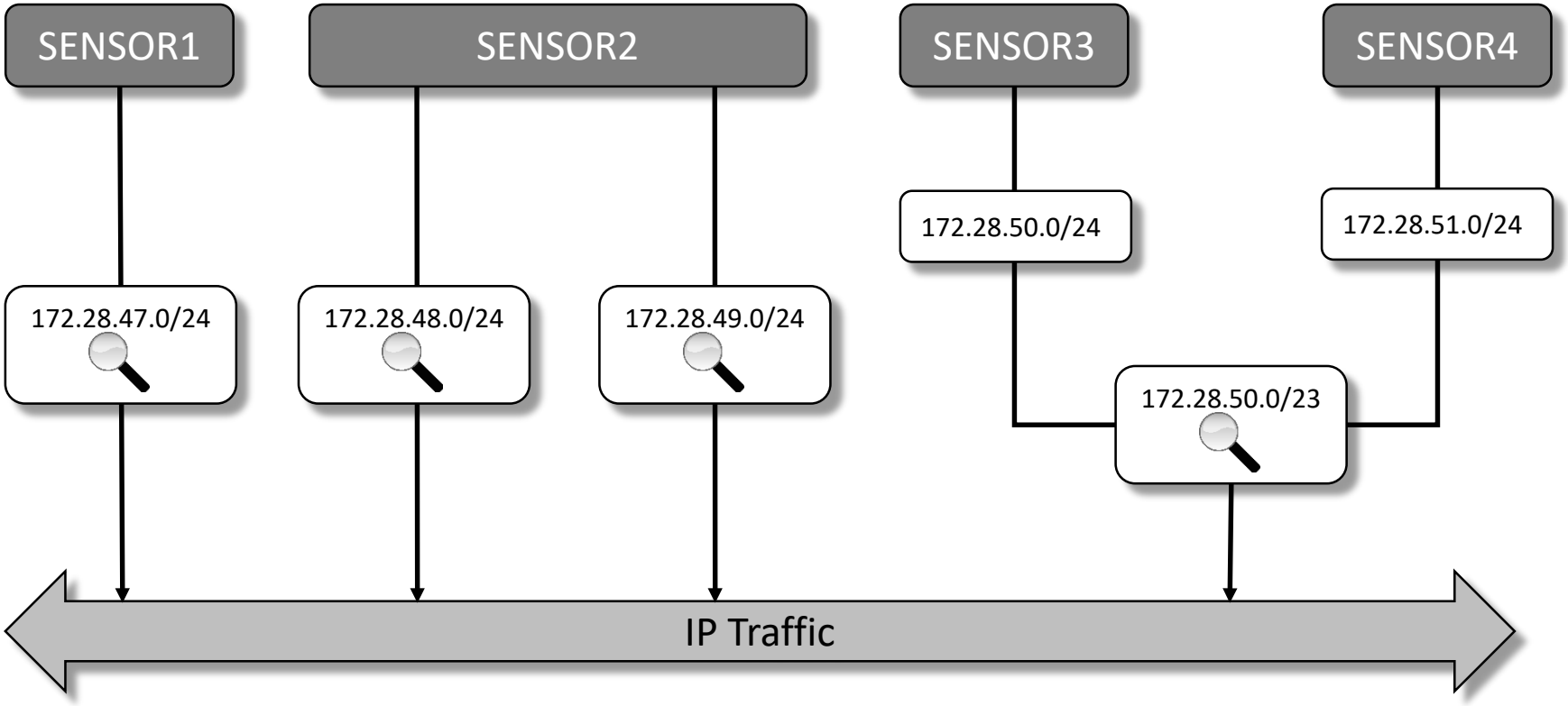
Members of one or more *classes*

Flexible configuration options

- single probe per sensor (common)
- multiple probes per sensor
- multiple sensors per probe



SiLK Sensors



Networks

Networks are logical address spaces that border a sensor

- used to assign *types* to network flow data

Internal

- space that is monitored

External

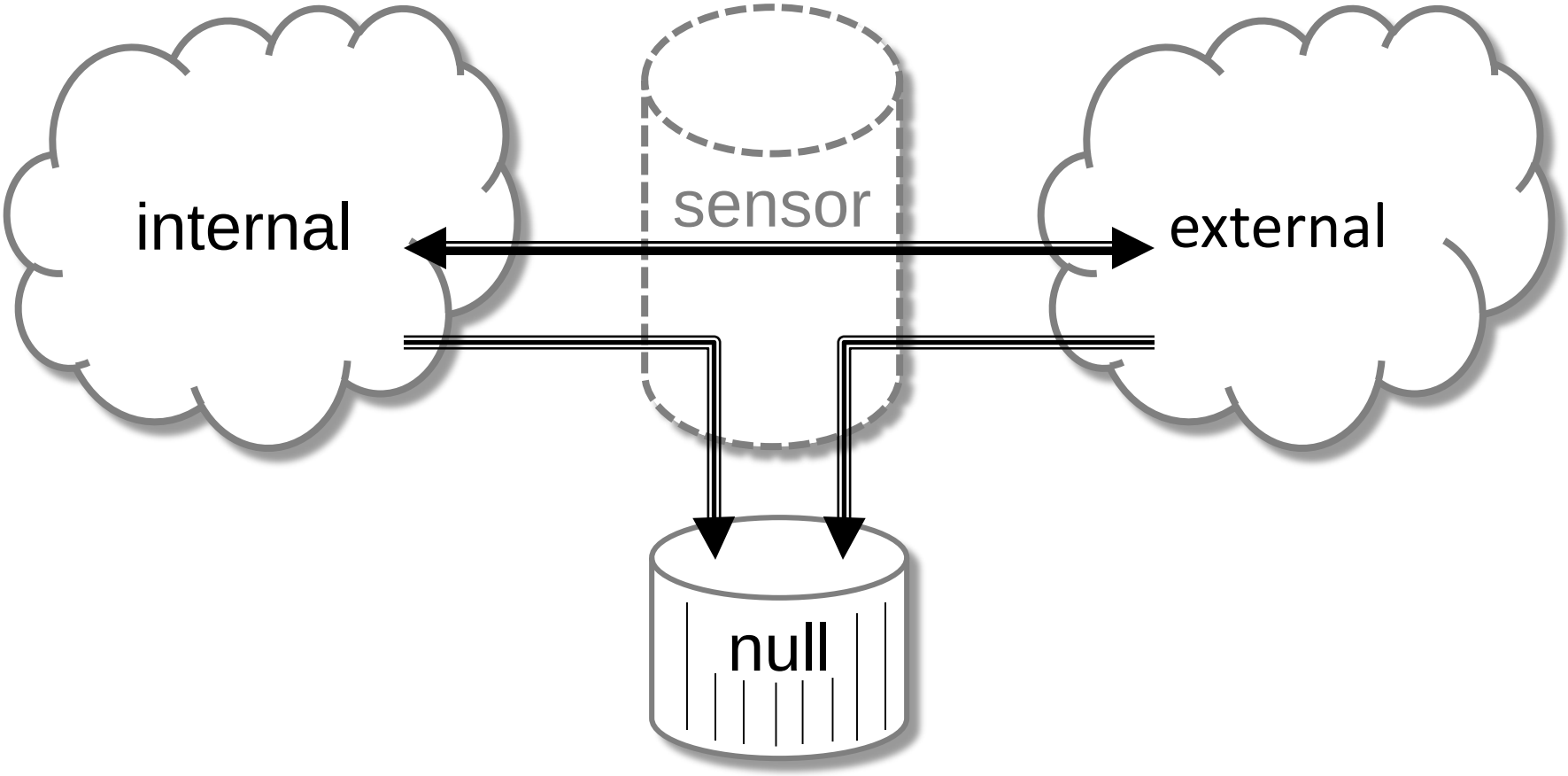
- space outside the monitored network

Null

- space that doesn't leave the sensor
- examples: blocked, BGP

internal
external
null

SiLK Networks



Classes

Classes represent network topological features

- label flow data relationship to infrastructure

Classes are comprised of

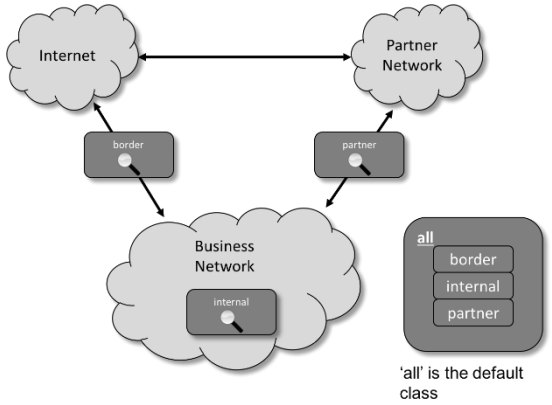
- unique names
- sensors
- types

Example classes

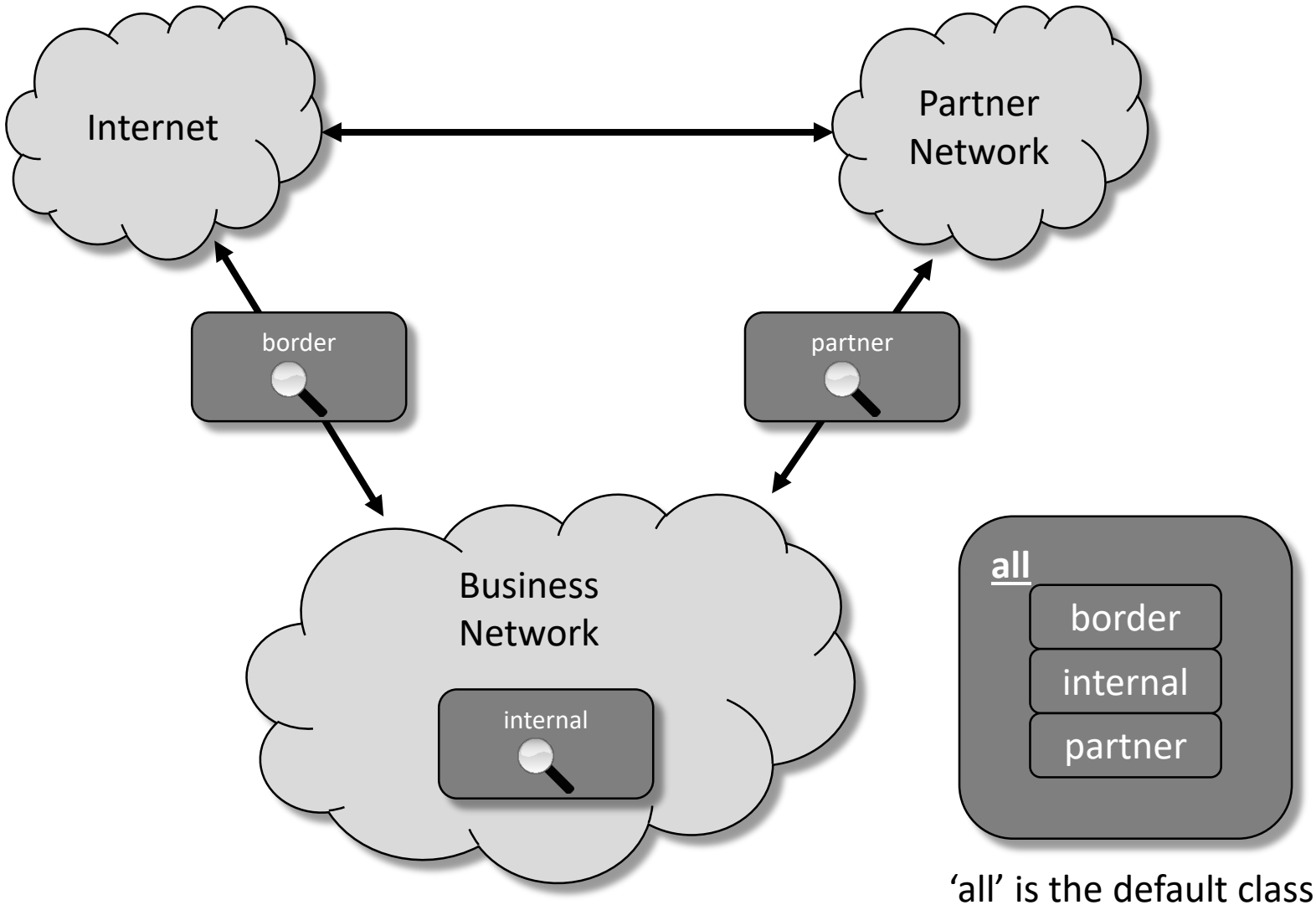
- “border”
- “internal”
- “partner”

One class must be defined

- ’all’ is default



SiLK Classes



'all' is the default class

Types Overview

Types describe packet movement between networks

External source to internal destination

- incoming

Internal source to external destination

- outgoing

Internal source to internal destination

- internal

External source to external destination

- external

Internal or external source that terminates at a sensor

- null

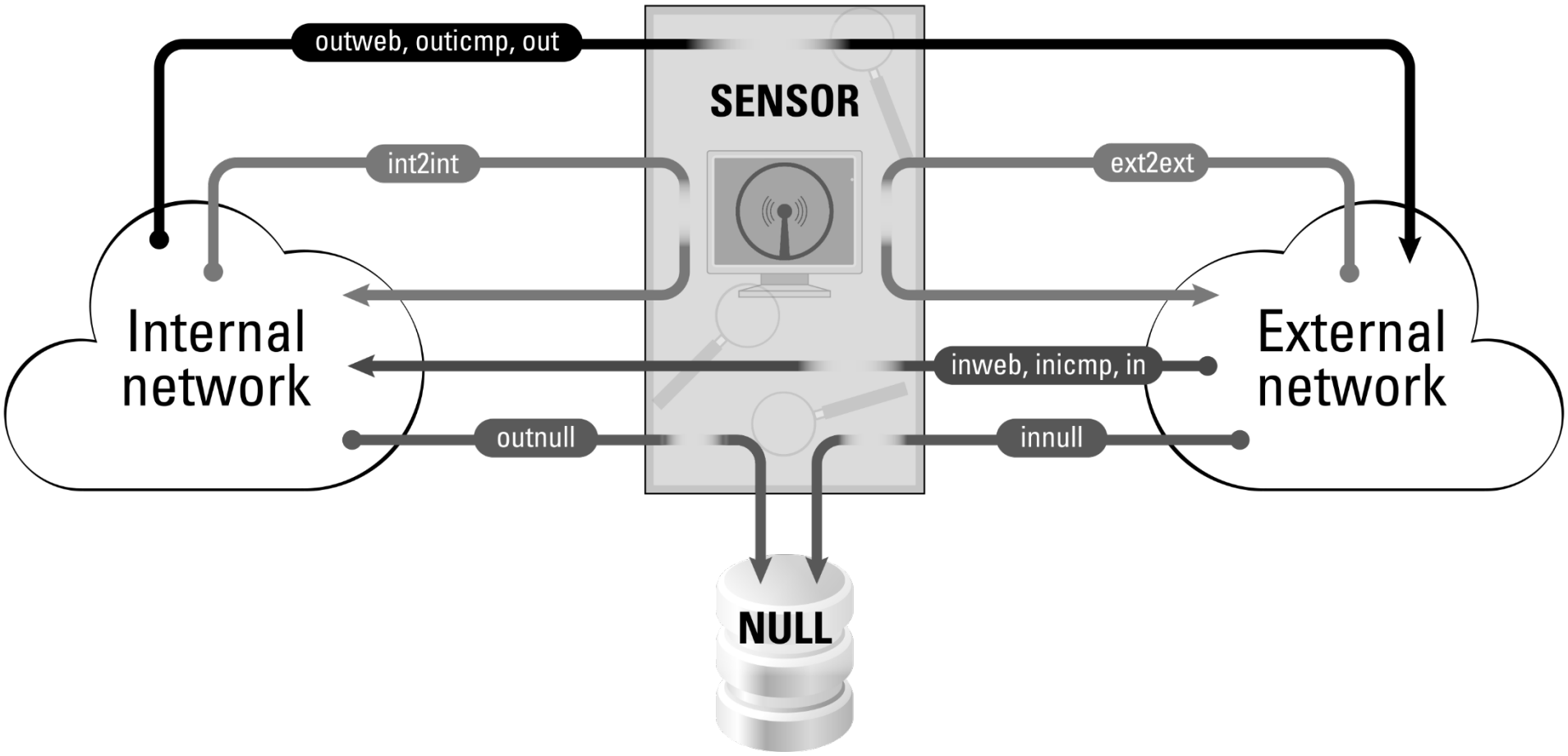
incoming
outgoing
internal
external
null

Types List

Type	Description
inweb, outweb	Incoming/outgoing TCP ports 80, 443
innull, outnull	Incoming/outgoing filtered traffic
inicmp, outicmp	Incoming/outgoing IP protocol 1
in, out	Incoming/outgoing not in above categories
int2int, ext2ext	Internal to internal, external to external
other	Source not internal or external, or destination not internal, external, or null

Names in **bold** are often default types

SiLK Types

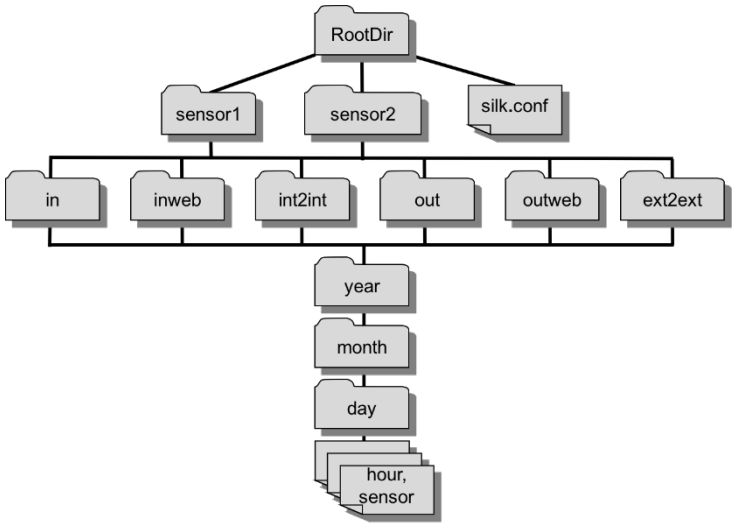


Repository Overview

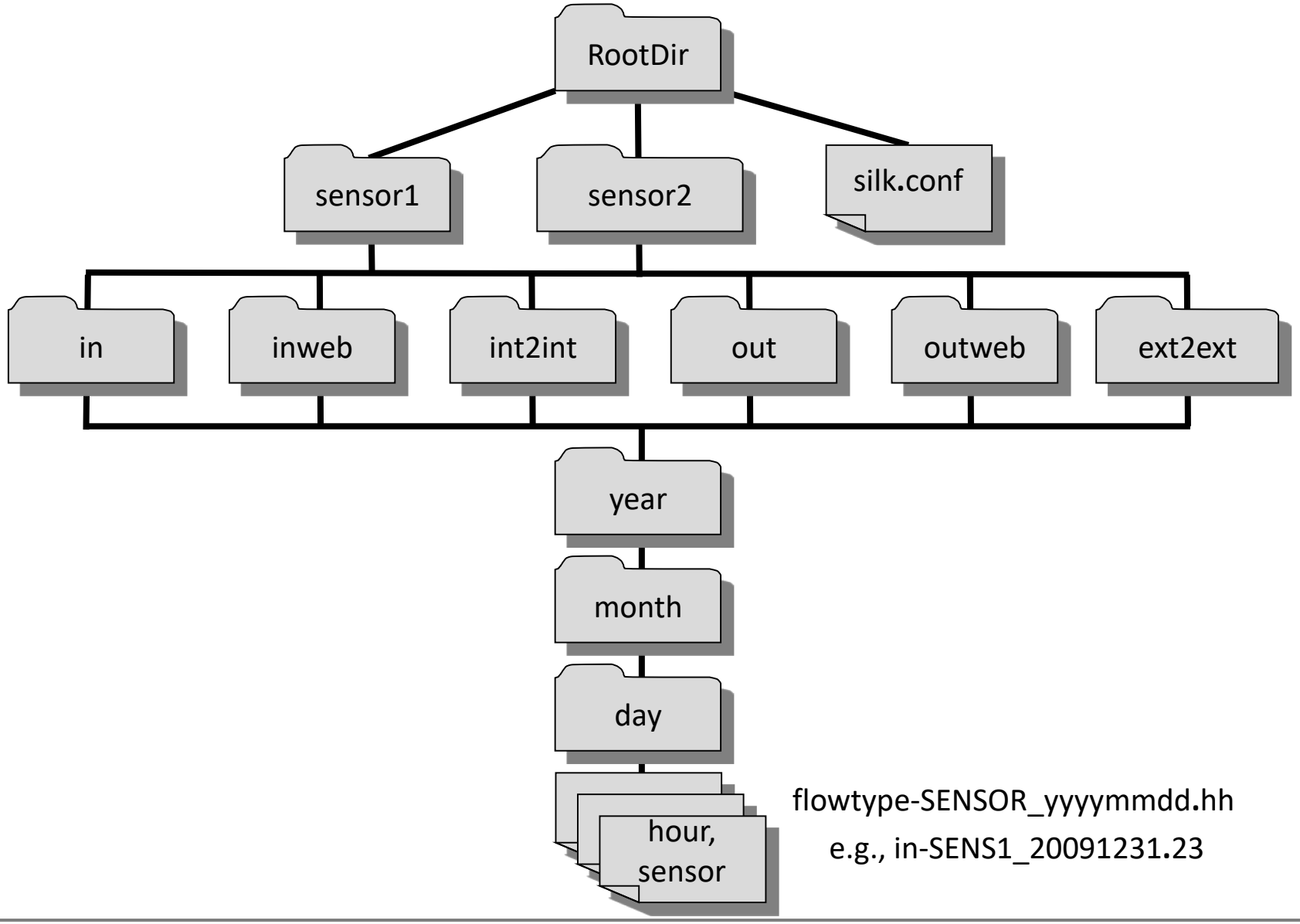
The *repository* is a hierarchical structure used to organize and store SiLK flow files

Combination of directories and files

- '/data' (default $ROOT directory)
- sensor (directory)
- type (directory)
- time (directory and files)



SiLK Repository



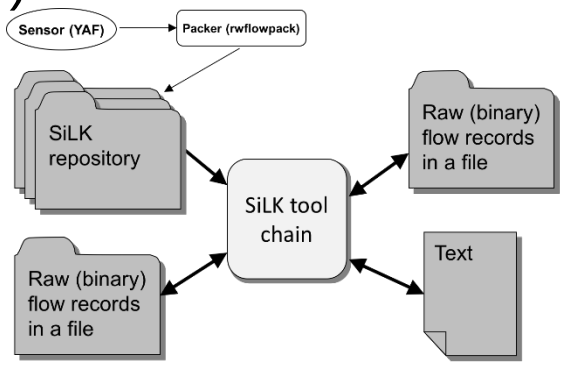
Analysis Suite

Read binary files

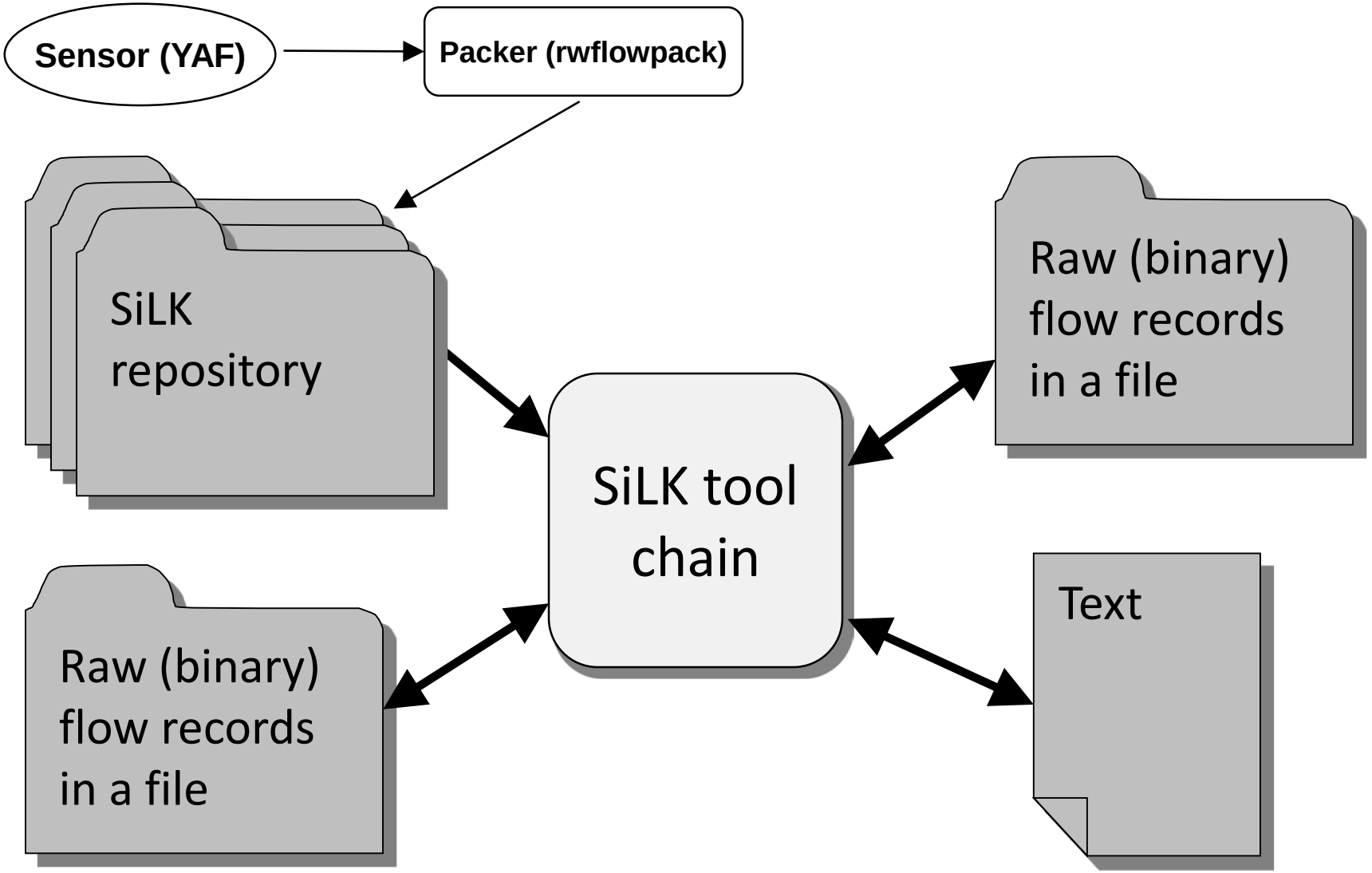
- SiLK
- other

Partition, sort, count, and display network flow records

- select network flow records using detailed criteria
- read or store SiLK binary formats
- read or store text
- read other formats (ex: pcap, IPFIX, NetFlow v5)



SiLK Analysis Suite



Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

rwsiteinfo

Print information from the site configuration files

rwsiteinfo Syntax

General form

```
rwsiteinfo --fields=fields[,fields…](required)
      --classes=class[,class…]
      --type=type[,type…]
      --flowtypes=class/type[,class/type…]
      --sensor=sensor[,sensor…]

rwsiteinfo --help
```

Example call

```
 rwsiteinfo --fields=type --sensor=T1
```

What Is This?

```
rwsiteinfo --fields=id-sensor --type=in
Sensor-ID|
        0|
        1|
```

```
rwsiteinfo --fields=id-sensor,default-type    \
--sensor=S0
Sensor-ID|Default-Type|
        0|           in|
        0|        inweb|
        0|      inicmp|
```

What Is This?

```
rwsiteinfo --fields=id-sensor --type=in
Sensor-ID|
        0|
        1|
```

Answer: Display sensor IDs that have ‘in’ types

```
rwsiteinfo --fields=id-sensor,default-type    \
--sensor=S0
Sensor-ID|Default-Type|
        0|          in|
        0|       inweb|
        0|      inicmp|
```

What Is This?

```
rwsiteinfo --fields=id-sensor --type=in
Sensor-ID|
        0|
        1|
```

Answer: Display sensor IDs that have ‘in’ repository types

```
rwsiteinfo --fields=id-sensor,default-type    \
--sensor=S0
Sensor-ID|Default-Type|
        0|           in|
        0|        inweb|
        0|      inicmp|
```

Answer: Display the sensor ID and default types for sensor S0

`rwsiteinfo` Helping the New Analyst

You have been asked to help a new analyst become familiar with the SOC SiLK deployment.

Using your knowledge of the `rwsiteinfo` command, help them identify the descriptive name and default SiLK types for sensor S1.

Review

Solution

```
rwsiteinfo --fields=describe-sensor,default-type \
--sensor=S1
```

Review

Exercise 1

This exercise will walk you through an analysis of the SiLK site configuraiton with the `rwsiteinfo` command:

- Understanding how SiLK is configured and the data each sensor is collecting will help analysts interpret output and perform configuration analysis.

Questions: 5

Time: 15 minutes

Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

rwfilter

Query data from the repository or raw file

`rwfilter` Overview

We will be covering the following basic parameters:

- `rwfilter` syntax - Input, Selection, Partition, Output, Other
- time
- simple numeric fields: ports, protocol, ICMP Type
- count of packets, bytes, duration
- specified IP addresses, CIDR blocks, and wildcards
- sets of IP addresses
- TCP Flags

rwfilter Syntax

General form

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

Example call

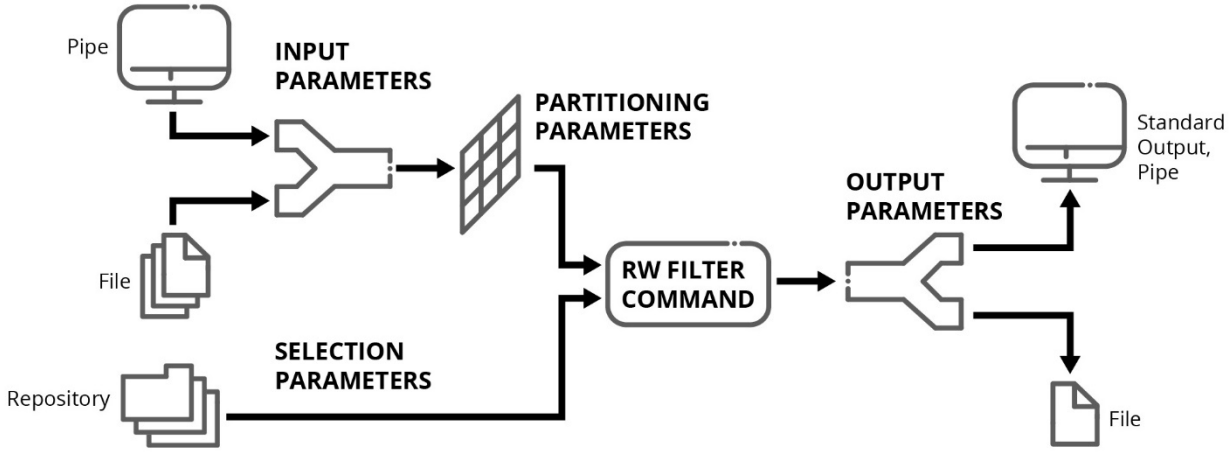
```
rwfilter --sensor=S0 --type=in \
  --start-date=2009/4/21T9 \
  --end-date=2009/4/21T16 \
  --protocol=0-255 --pass=workday-21.rw
```

`rwfilter` Command Structure

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.



`rwfilter` Selection and Input Criteria

Selection options control access to repository files:

- `--start-date=2009/4/21:00`
- `--end-date=2009/4/21T03` (ISO format)
- `--sensor=S0`
- `--class=all`
- `--type=in,inweb`

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

- `myfile.rw`
- `stdin or -`
- useful for chaining filters through stdin/stdout

`rwfilter` Partitioning by Time (Raw File)

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

- `myfile.rw`
- `stdin` or `-`

```
--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.

```
--stime=2009/4/21T13:00-2009/4/21T13:29          # ½ hr
--etime=2009/4/21T13:00:00-2009/4/21T13:00:09    # 10 sec
--stime=2009/4/21T13:00-2009/4/21T13:00:48.725 # 48.725s
```

`rwfilter` Output Criteria

`rwfilter` leaves the flows in binary (compact) form.

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

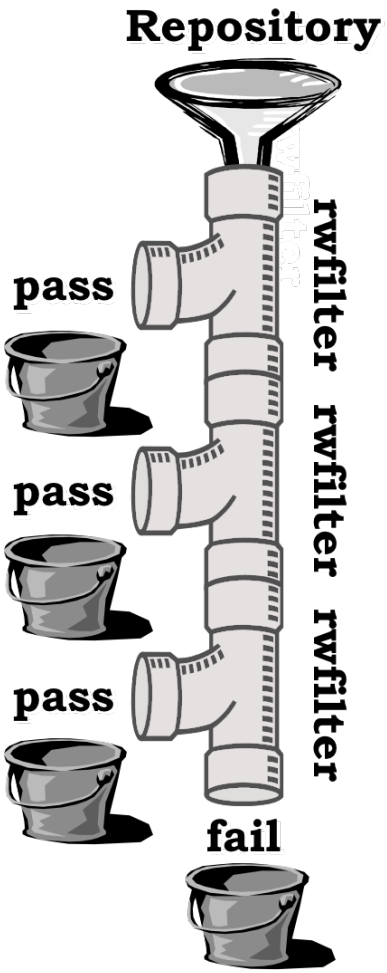
Other useful output

- `--print-filenames`
  `--print-missing-files`
- `--print-statistics` or
  `--print-volume-statistics`

`rwfilter` 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=2014/11/22 --type=out,outweb \
--duration=300- --pass=stdout \
--fail=outbound_less_then_5min.rw | \
rwfilter --input-pipe=stdin --bytes=40000000- \
--pass=large_outbound.rw --fail=small_outbound.rw
```

`rwfilter` 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
```

`rwfilter` Packets, Bytes, and Duration

```
--packets=                  # packets in the flow
--bytes=                    # bytes in the packets in flow
--bytes-per-packet=         # average
--duration=                 # Time between stime - etime

--packets=3-
--bytes=40-570
--bytes-per-packet=40.0-75.125
--duration=120-
```

What Is This?

```
rwfilter \
      --start-date=2010/12/08 \
      --type=outweb \
      --bytes=100000- \
      --pass=stdout \
| rwfilter \
      --input-pipe=stdin \
      --duration=60- \
      --pass=long-http.rw \
      --fail=short-http.rw
```

What Is This?

```
rwfilter \
      --start-date=2010/12/08 \
      --type=outweb \
      --bytes=100000- \
      --pass=stdout \
| rwfilter \
      --input-pipe=stdin \
      --duration=60- \
      --pass=long-http.rw \
      --fail=short-http.rw
```

Answer: Classifies 100,000+-byte web output flows by fast or slow transfer

`rwfilter` Policy Violation

On 2014/12/12, the CIO put out a policy change on what servers are authorized to provide remote desktop access.

Using your knowledge of `rwfilter`, generate a SiLK raw file that contains network flows with RDP servers and internet clients from 2014/12/26 through 2014/12/28.

Review

Solution

```
rwfilter --start-date=2014/12/26 \
--end-date=2014/12/28 --protocol=6 \
--type=out --sport=3389 --packets=3- \
--pass=rdp.rw
```

Review

rwfileinfo

Display a variety of characteristics for each file format produced by the SiLK tool suite

rwfileinfo Syntax

General form

```
rwfileinfo file.rw --fields=field[,field…]
```

Example call

```
 rwfileinfo bad_IP.rw --fields=version
```

`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

```
$ rwfilter --sensor=S0 --type=in,out --start-date=2009/04/21T15 \
--protocol=1 --pass=icmprecords.rw

$ rwfileinfo icmprecords.rw
icmprecords.rw:
format(id)          FT_RWIPV6ROUTING(0x0c)
version             16
byte-order          littleEndian
compression(id)     none(0)
header-length       176
record-length       88
record-version      1
silk-version        3.16.0
count-records       39
file-size           3608
command-lines     1  rwfilter --sensor=S0 --type=in,out
--start-date=2009/04/21T15 --protocol=1 --pass=icmprecords.rw
```

rwsort

Sort SiLK flow records on one or more fields

rwsort Syntax

General form

```
rwsort --fields=<fields> [switches] [files]
```

Example call

```
rwsort t20.rw --fields=stime \
   --output-path=t20bystime.rw
```

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

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 are stored in /tmp by default.

```
rwsort t20.rw --fields=sip,sport,dport \
--temp-directory=/user/home/t20sort.rw
```

`rwsort` Data analysis

You have saved an rwfilter query that contains flow records for a data exfiltration investigation. The SiLK raw file name is `badip.rw` and it will be used for a time series analysis.

Using your knowledge of `rwsort`, sort the records in this raw file by source IP address, start time, and save it as `badip_sorted.rw`.

Review

Solution

```
rwsort badip.rw --fields=sip,stime > badip_sorted.rw
```

Review

rwcut

rwcut

Print selected fields of binary SiLK flow records to a screen

`rwcut` Syntax

General form

```
rwcut [switches] [file]
```

Example call

```
rwcut bad_IP.rw --fields=sip,dip,bytes,stime
```

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.

`rwcut` Default Display

By default

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

`--all-fields`

`rwcut` Print flow records

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

```
            sIP|            dIP|pro|pkts|bytes|
207.240.215.71| 128.3.48.203|  1|   1|   60|
207.240.215.71|  128.3.48.68|  1|   1|   60|
207.240.215.71|  128.3.48.71|  1|   1|   60|
```

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 the above three options
- `--legacy-timestamps`: better for import to Excel

`rwcut` Data analysis

After sorting the `badip_sorted.rw` raw file by source IP address and start time, an analyst wants to display the data in a popular graphical tool that only accepts text files with the following format:

```
SIP%DIP%BYTES
```

Using your knowledge of `rwcut`, save the records in the required text format.

Review

Solution

```
rwcut badip_sorted.rw --fields=sip,dip,bytes \
--delimited='%' > badip_sorted.txt
```

Review

Exercise 2

This exercise focuses on the following SiLK commands:

- `rwfileinfo`, `rwcut`, `rwsort`, and the all-powerful `rwfilter`

Time: 30 minutes

Questions: 7

Beginning Analysis with SiLK

Introduction to SiLK

Network flow

SiLK components

Repository tools

Basic record handling tools

Basic profiling tools

Summary

`rwcount`, `rwstats`, `rwuniq` Syntax

`rwcount`: Count volume across time periods.

`rwstats`: Count volume across IP addresses, port, or protocol and create descriptive statistics.

`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`, `rwstats`, `rwuniq` Applications

Count [volume] by [key field] and print [summary]:

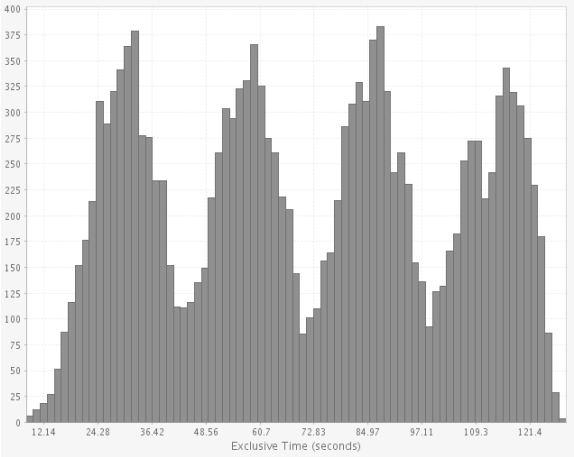
- basic bandwidth study
  - Count bytes by hour and print the results.
- top-10 talkers list
  - Count bytes by source IP and print the 10 highest IP addresses.
- user profile
  - Count records by dIP-dPort pair and print all the pairs.
- potential scanners
  - Count unique dIPs by sIP and print the sources that contacted more than 100 destinations.

rwcount

rwcount

Print traffic summaries across time

rwcount Syntax



General form

```
rwcount [switches] [file]
```

Example call

```
 rwcount bad_IP.rw --bin-size=600
```

Counts records, bytes, and packets by **time** and displays the 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 software

`rwcount` Time Bins

rwcount uses a 30 second default time bin

Other time bins must be specified in seconds with the **bin-size** switch

`--bin-size=86400` # 24 hour bins

rwcount Summary

The bin key is always time. You choose the period.

The aggregate measures are chosen for you. They are flows/records, bytes, and packets.

```
rwfilter --sensor=S0 --start=2009/4/21 \
    --type=in --proto=1 --pass=stdout  \
 | rwcount --bin-size=3600
                Date|Records|   Bytes|Packets|
...
2009/04/21T13:00:00|  10.00| 2460.00|  41.00|
2009/04/21T14:00:00|  29.00| 8036.00| 107.00|
2009/04/21T15:00:00|  22.00| 2214.00|  47.00|
2009/04/21T16:00:00|  10.00| 1586.00|  23.00|

...
```

What Is This?

```
rwcount MSSP.rw --bin-size=3600
                Date|    Records|        Bytes|    Packets|
2010/12/08T00:00:00| 1351571.66| 73807086.40| 1606313.61|
2010/12/08T01:00:00| 1002012.43| 54451440.59| 1185143.62|
2010/12/08T02:00:00| 1402404.61| 77691865.26| 1675282.27|
2010/12/08T03:00:00| 1259973.65| 68575249.90| 1491393.08|
2010/12/08T04:00:00|  939313.56| 51410968.24| 1118584.81|
2010/12/08T05:00:00|  459564.75| 80862273.32| 1742058.62|
2010/12/08T06:00:00| 1280651.23| 69881126.41| 1519435.24|
```

What Is This?

```
rwcount MSSP.rw --bin-size=3600
                Date|    Records|       Bytes|    Packets|
2010/12/08T00:00:00| 1351571.66| 73807086.40| 1606313.61|
2010/12/08T01:00:00| 1002012.43| 54451440.59| 1185143.62|
2010/12/08T02:00:00| 1402404.61| 77691865.26| 1675282.27|
2010/12/08T03:00:00| 1259973.65| 68575249.90| 1491393.08|
2010/12/08T04:00:00|  939313.56| 51410968.24| 1118584.81|
2010/12/08T05:00:00|  459564.75| 80862273.32| 1742058.62|
2010/12/08T06:00:00| 1280651.23| 69881126.41| 1519435.24|
```

Answer: Summarize the MSSP.rw file in one hour bins.

`rwcount` Bin Traffic

Your supervisor notices that there was a disturbance in the force from December 29, 2014 through January 2, 2015. Using your knowledge of `rwfilter` and `rwcount`, what command would help determine which 30 minute interval had the most outgoing traffic on sensor DS for this time period?

Review

Solution

```
rwfilter --start-date=2014/12/29 \
--end-date=2015/01/02 --sensor=DS \
--protocol=0- --type=out,outweb \
--pass=stdout | rwcount --bin-size=1800
```

Review

rwstats

rwstats

Print top-N or bottom-N lists or summarize data by protocol

rwstats Syntax

General form

```
rwstats [switches] [file]
```

```
--fields=KEY --value=VOLUME
--count=N or --threshold=N or
--percentage=N
[--top or --bottom]
```

Example call

```
rwstats bad_IP.rw --fields=sip --count=10
```

- Choose one or two key fields.
- Count one of the records, bytes, or packets.
- great for top-N lists and count thresholds

What Is This?

```
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
                 sIP|      Bytes|    %Bytes|   cumul_%|
         71.55.40.62| 1754767148| 41.541935| 41.541935|
        71.55.40.169| 1192063164| 28.220617| 69.762552|
        71.55.40.179|  331310772|  7.843372| 77.605923|
        71.55.40.204|  170966278|  4.047415| 81.653338|
      177.249.19.217|  122975880|  2.911301| 84.564639|
         71.55.40.72|  110726717|  2.621318| 87.185957|
        71.55.40.200|  101593627|  2.405103| 89.591060|
      177.71.129.255|   40166574|  0.950894| 90.541954|
         71.55.40.91|   35316554|  0.836076| 91.378030|
     149.249.114.204|   26634602|  0.630541| 92.008571|
```

What Is This?

```
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
                 sIP|      Bytes|    %Bytes|   cumul_%|
         71.55.40.62| 1754767148| 41.541935| 41.541935|
        71.55.40.169| 1192063164| 28.220617| 69.762552|
        71.55.40.179|  331310772|  7.843372| 77.605923|
        71.55.40.204|  170966278|  4.047415| 81.653338|
      177.249.19.217|  122975880|  2.911301| 84.564639|
         71.55.40.72|  110726717|  2.621318| 87.185957|
        71.55.40.200|  101593627|  2.405103| 89.591060|
      177.71.129.255|   40166574|  0.950894| 90.541954|
         71.55.40.91|   35316554|  0.836076| 91.378030|
     149.249.114.204|   26634602|  0.630541| 92.008571|
```

Answer: Display the Top 10 source IP addresses by byte volumes.

`rwstats` Top Protocols

Your supervisor provides you a raw file (inbound.rw) containing all inbound traffic. Which SiLK command would display the top-10 incoming protocols used based on bytes?

Review

Solution

```
rwstats inbound.rw --fields=protocol \
--bytes --count=10
```

Review

rwuniq

rwuniq

Bin SiLK flow records by a key and print each bin's volume

rwuniq Syntax

General form

```
rwuniq --fields=[switches] [file]
```

Example call

```
 rwuniq bad_IP.rw --fields=sip,dip \
 --values=bytes
```

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

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

`rwuniq` Counting Options

Key	Volume	Summary

`--fields=KEYS`
 	
`--bin-time=SECS`
	`--value={`
	`flows | bytes |`
	`packets |`
	`sip-distinct |`
	`dip-distinct |`
	`stime | etime}…`
		`--sort-output`
		`--VOLUME=MIN`
		`--VOLUME=MIN-MAX`

KEYS is any valid specification of SiLK fields:

- `rwuniq --fields=sIP,sPort,sTime --bin-time=60`
- `rwuniq --fields=1-5`

Choose *any* combination of volumes or `--all-counts` for all.

Use `--sort-output` to sort by *key*, not by volume (no top-N lists).

What Is This?

```
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

           dIP|sPort |     Bytes|Packets| Records|     sTime-Earliest|       eTime-Latest|
 12.113.41.190|   80 |     12782|     20|       4|2010/12/08T18:42:51|2010/12/08T18:58:49|
30.182.228.143|   80 | 203907933| 143611|       2|2010/12/08T18:53:59|2010/12/08T19:01:47|
 37.153.24.229|   80 | 205628625| 144829|       2|2010/12/08T18:29:11|2010/12/08T18:42:51|
 82.180.203.87|   80 | 213013145| 150896|      92|2010/12/08T18:06:36|2010/12/08T18:32:33|
82.180.203.197|   80 |       800|      8|       2|2010/12/08T18:43:30|2010/12/08T18:43:30|
88.124.166.233|   80 | 223930369| 158276|      97|2010/12/08T18:08:55|2010/12/08T18:32:25|
88.124.166.233|  443 |    509285|    732|      43|2010/12/08T18:06:57|2010/12/08T18:51:11|
94.239.226.247|   80 | 124833037|  96047|       3|2010/12/08T18:25:22|2010/12/08T19:21:34|
  109.95.61.80|   80 |   8467397|   6325|      90|2010/12/08T18:08:59|2010/12/08T18:10:09|
  139.65.186.4|   80 | 204123360| 143794|       3|2010/12/08T18:19:48|2010/12/08T18:26:36|
139.177.10.136|   80 | 407978375| 287354|       6|2010/12/08T18:20:03|2010/12/08T19:01:30|
198.237.16.172|   80 | 159066748| 112025|       1|2010/12/08T18:18:43|2010/12/08T18:46:55|
219.149.72.154| 1024 |        44|      1|       1|2010/12/08T18:50:40|2010/12/08T18:50:40|
249.216.88.172|   80 |        88|      2|       2|2010/12/08T18:44:42|2010/12/08T18:44:47|
250.211.100.88|   80 |   3295160|   2492|      42|2010/12/08T18:47:50|2010/12/08T18:58:53|
```

What Is This?

```
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

           dIP|sPort |     Bytes|Packets| Records|     sTime-Earliest|       eTime-Latest|
 12.113.41.190|   80 |     12782|     20|       4|2010/12/08T18:42:51|2010/12/08T18:58:49|
30.182.228.143|   80 | 203907933| 143611|       2|2010/12/08T18:53:59|2010/12/08T19:01:47|
 37.153.24.229|   80 | 205628625| 144829|       2|2010/12/08T18:29:11|2010/12/08T18:42:51|
 82.180.203.87|   80 | 213013145| 150896|      92|2010/12/08T18:06:36|2010/12/08T18:32:33|
82.180.203.197|   80 |       800|      8|       2|2010/12/08T18:43:30|2010/12/08T18:43:30|
88.124.166.233|   80 | 223930369| 158276|      97|2010/12/08T18:08:55|2010/12/08T18:32:25|
88.124.166.233|  443 |    509285|    732|      43|2010/12/08T18:06:57|2010/12/08T18:51:11|
94.239.226.247|   80 | 124833037|  96047|       3|2010/12/08T18:25:22|2010/12/08T19:21:34|
  109.95.61.80|   80 |   8467397|   6325|      90|2010/12/08T18:08:59|2010/12/08T18:10:09|
  139.65.186.4|   80 | 204123360| 143794|       3|2010/12/08T18:19:48|2010/12/08T18:26:36|
139.177.10.136|   80 | 407978375| 287354|       6|2010/12/08T18:20:03|2010/12/08T19:01:30|
198.237.16.172|   80 | 159066748| 112025|       1|2010/12/08T18:18:43|2010/12/08T18:46:55|
219.149.72.154| 1024 |        44|      1|       1|2010/12/08T18:50:40|2010/12/08T18:50:40|
249.216.88.172|   80 |        88|      2|       2|2010/12/08T18:44:42|2010/12/08T18:44:47|
250.211.100.88|   80 |   3295160|   2492|      42|2010/12/08T18:47:50|2010/12/08T18:58:53|
```

Answer: Display (in sorted order) the destination IP addresses that source IP address 71.55.40.62 communicated with and the source ports it used

`rwuniq` Scanner

You generate the `scanning.rw` raw file to analyze outbound scanning activity originating from your network. Using your knowledge of `rwuniq`, how would you identify the internal IP addresses that connected to at least 400 distinct internet IP addresses, and then summarize the records and bytes for each?

Review

Solution

```
rwuniq scanning.rw --fields=sip \
--dip-distinct=400 --flows --bytes
```

Review

`rwuniq` Versus `rwstats`

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

IPset Tools

A tool suite that allows you to generate and manipulate IPset files:

- `rwset`
- `rwsetcat`
- `rwsetmember`
- `rwsetbuild`
- `rwsettool`

IPset Tools

An `IPset` is a mathematical set of IP addresses stored as a binary data structure

`rwsetbuild`: creates IPsets from text

`rwset`: creates IPsets from binary flow records

`rwsetcat`: displays an IPset as text

`rwsetmember`: tests if an address is in given IPsets

`rwsettool`: performs IPset algebra (intersection, union, set difference) on multiple IPset files

CERT SiLK IPset tools available in an independent distribution

`IPset` Applications

Too many addresses for the command line?

- Blacklist (spam networks, IP IoCs, etc)
- Whitelist (mail servers, VPNs, etc)
- arbitrary list of any type of addresses

Create an IPset!

- 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`

`rwset` Syntax

General form

```
rwset <IPset-Creation-Switch> [switches] [file]
```

Example call

```
 rwset badip.rw --dip-file=badip_dip.set
```

Read SiLK flow records and generate one or more binary IPset files.

At least one creation switch must be specified, and only one IPset of each possible type may be created.

`rwsetcat` and `rwsetmember` Syntax

General form

```
rwsetcat [switches] [IPset file]
```

Example call

```
rwsetcat BadIP.set --network-structure=/24
```

By default, prints the IP address of the specified IPset file on the screen

General form

```
rwsetmember [switches] [input_set]
```

Example call

```
rwsetmember BadIP.set 192.168.1.1 --count
```

Determine the existence of IP address in one or more IPset files.

What Is This?

```
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
```

What Is This?

```
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
```

Answer: Group the distinct destination IP addresses for ‘out’ type flows and count them by /16 network.

`rwsetbuild` Syntax

General form

```
rwsetbuild [switches] [input file] [output file]
```

Example call

```
 cat ipset.txt
 192.168.0.0/24
 10.10.10.11–10.10.10.29
 rwsetbuild ipset.txt ipset.set --ip-ranges
```

Read IP addresses from a text input file and write a binary IPset file to an output file.

`rwsettool` Syntax

General form

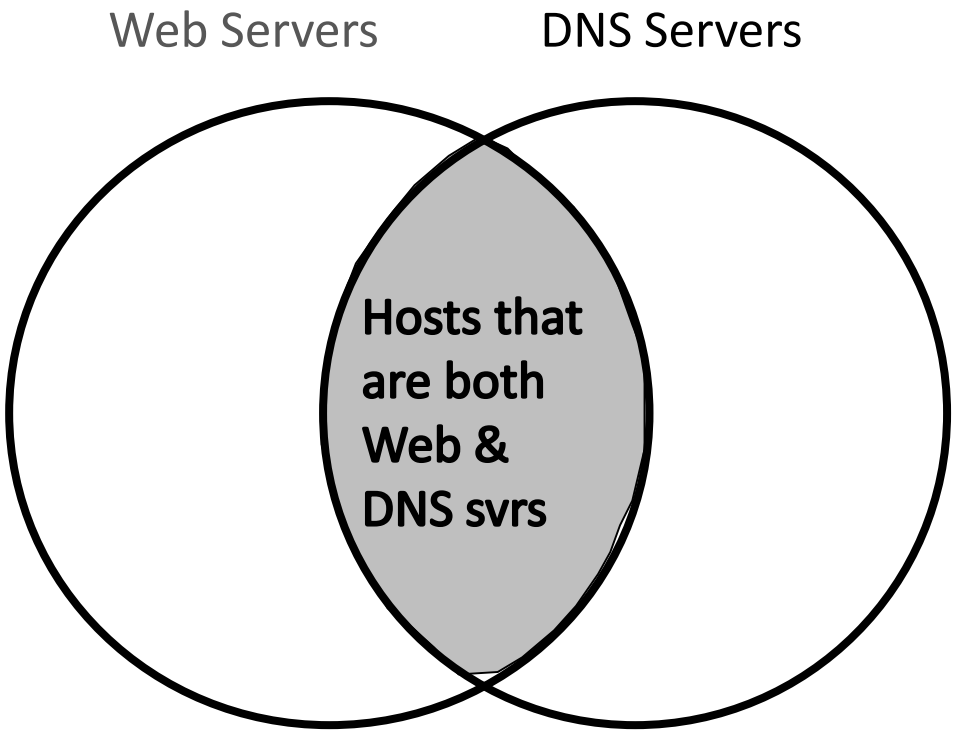
```
rwsettool operation [switches] [IPset file]
```

Example call

```
 rwsettool --difference web.set    \
 dns.set --output-path=web_not_dns.set
```

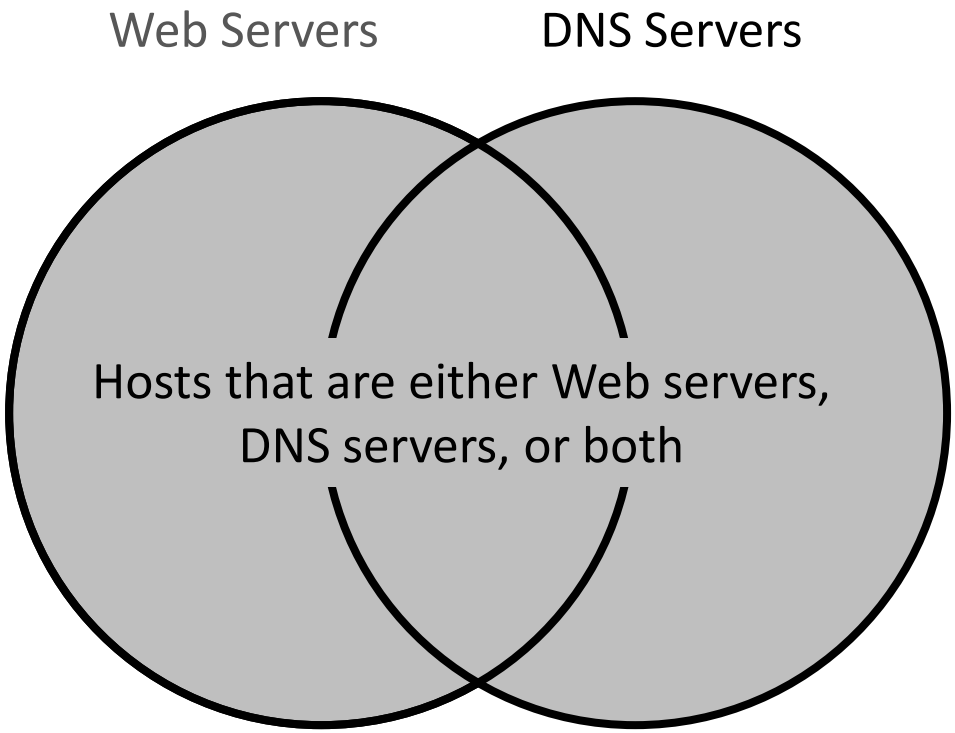
Performs the specified operation on one or more IPsets.

Set Intersection



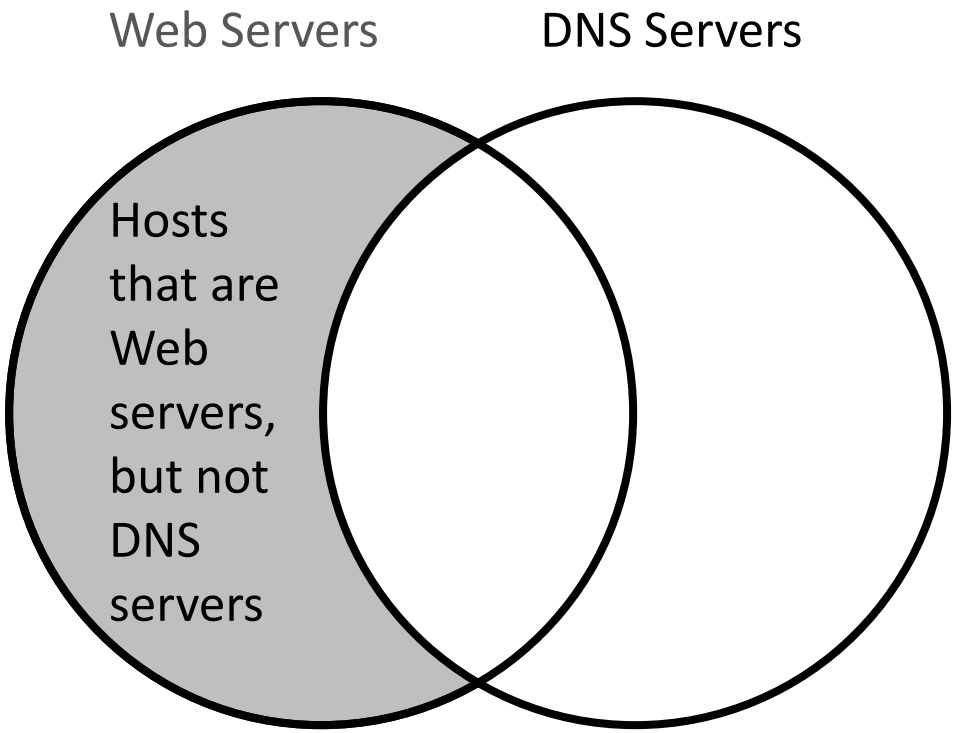

```
rwsettool --intersect  web.set  dns.set \
--output-path=web_and_dns.set
```

Set Union




```
rwsettool --union  web.set  dns.set \
--output-path=web_or_dns.set
```

Set Difference




```
rwsettool --difference  web.set  dns.set \
--output-path=web_not_dns.set
```

What Is This?

```
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

     |        Recs|      Packets|          Bytes|Files|
Total|    30767188|     81382782|    35478407950|   48|
 Pass|    26678669|     31743084|     1464964676|     |
 Fail|     4088519|     49639698|    34013443274|     |

rwset --sip-file=MSSPsource.set MSSP.rw

rwsettool --intersect MSSP.set MSSPsource.set \
   --output=activeMSSP.set

rwsetcat --count-ips activeMSSP.set

22
```

What Is This?

```
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

     |        Recs|      Packets|          Bytes|Files|
Total|    30767188|     81382782|    35478407950|   48|
 Pass|    26678669|     31743084|     1464964676|     |
 Fail|     4088519|     49639698|    34013443274|     |

rwset --sip-file=MSSPsource.set MSSP.rw

rwsettool --intersect MSSP.set MSSPsource.set \
   --output=activeMSSP.set

rwsetcat --count-ips activeMSSP.set

22
```

Answer: Count the number of IP addresses from two /24 networks that were active on 2010/12/08.

Exercise 3

This exercise focuses on the following SiLK commands:

- `rwcount`, `rwstats`, `rwuniq`, `rwsetbuild`, `rwset`, `rwsetcat`, `rwsetmember`, `rwsettool`, and again the all-powerful `rwfilter`

Time: 45 minutes

Questions: 6

Furthering Your SiLK Analysis Skills - 1

Each tool has a `--help` option.

SiLK Reference Guide, SiLK Analysts' Handbook

- Both are available on the SiLK tools website:
  - http://tools.netsa.cert.org

Email support

- netsa-help@cert.org

Furthering Your SiLK Analysis Skills - 2

Tool tips

- SiLK Tooltips link on http://tools.netsa.cert.org

Flow analysis research and advanced techniques

- http://www.cert.org/flocon
- http://www.cert.org/netsa

Questions?

Contact Information

DHS Shared Cybersecurity Services

Arlene Guevara Zuleta

(703) 235-4920

Arlene.Guevara-Zulet@hq.dhs.gov

SEI/CERT

NetSA Mentor

netsa-mentor@cert.org

Geoffrey Sanders

703-247-1393

gtsanders@sei.cmu.edu

Nathaniel Richmond

703-247-1395

nr@sei.cmu.edu