Analysis Pipeline

Streaming flow analysis with alerting

Dan Ruef - SEI

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JAN 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**
4. TITLE AND SUBTITLE **Analysis Pipeline: Streaming flow analysis with alerting**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
FloCon 2011, in Salt Lake City, Utah, on January 10-13, 2011.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**33**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Something Completely Different

IPFIX Interop Meeting

Prague, Czech Republic

March 24-26, 2011

Before the IETF meeting

The EU Seventh Framework DEMONS project is organizing an IPFIX Interoperability Event to be held immediately preceding the IETF 80 meeting in Prague, Czech Republic, on March 24-26, 2011. Implementors of products exporting or collecting network flow data with IPFIX will meet at the event to test the interoperability of their products against other implementations.

More details to follow on the DEMONS website; questions can be directed to the interop organizer, Brian Trammell, trammell@tik.ee.ethz.ch.

Agenda

Moves analyses from retroactive to real time

Pipeline capabilities

Stages of pipeline

Streaming analysis coding issues

SiLK

SiLK was built to effectively query a repository

- Everything is retroactive

Issues with time groupings

- Easy to analyze each hour
- Difficult to investigate every 1 hour period

Need many SiLK commands to isolate a value

Closest to real time is batched jobs

Pipeline

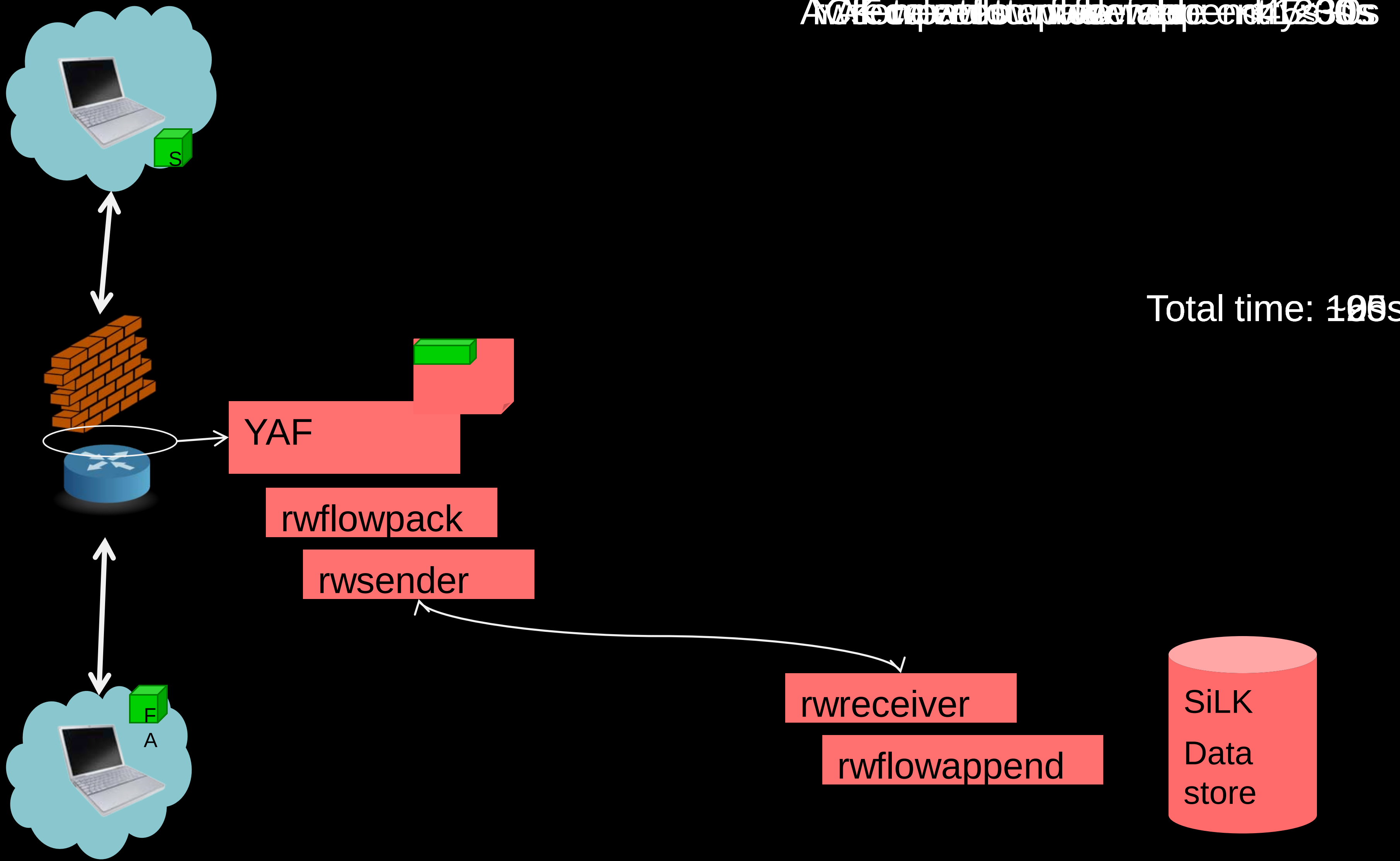
Pipeline is a single program, coded in C

- Configurable filters, evaluations, and alerting
- Parameters are read from a config file at startup
- Any number of filters and evaluations

Analyzes flow records en route to repository

- Processes data one flow file at a time
- Builds and keeps state between the files

Mechanics of Flow Collection



Pipeline Timing

Uses latest flow end time from each file to keep time and timestamp data

Sliding window time based analysis

- Keeps records in state for specified time duration
- Analyzes every time period not mutually exclusive time period blocks

Simple evaluation example:

- Alert if more than X bytes are sent in 5 minutes

Capabilities

Finite State Beacon Detection

Sensor Outage Detection

IPv6 Tunnel Detection

Passive FTP Detection

Watchlists

Flow counts

Flow field based capabilities (Can be combined)

- Sum or Average of the field value (bytes, packets, durations, etc)
- Proportion of flows with a given field value (TCP, Web, etc)

Flow Path

Filters
- All flows go through each filter
- Filter based on any field in flow record

Evals
- Filtered flows passed to associated eval
- Time sensitive state kept here

Alerts
- Alerts created when eval thresholds met
- Can be rate-limited

Filters

Stateless and need no concept of time

- Very low cost on time and memory

Role is to send only pertinent flows to evals

Stores list of flows that pass filter

- Deletes them after evaluations and alerts finish

Try to mimic features of rwFilter

Filters

All flow records are sent through each filter independently.

Operators for any field in flow record

- <, <=, >, >=, ==, !=, IN_LIST, NOT_IN_LIST
- Each filter can have multiple “anded” comparisons

IN_LIST and NOT_IN_LIST work on two types of lists

- User defined comma-separated lists, e.g. [1, 2, 3, 4, 5…]
- Ipset files: Overwriting the file allows pipeline to update the list

Different fields in flows can be compared

- sport < dport

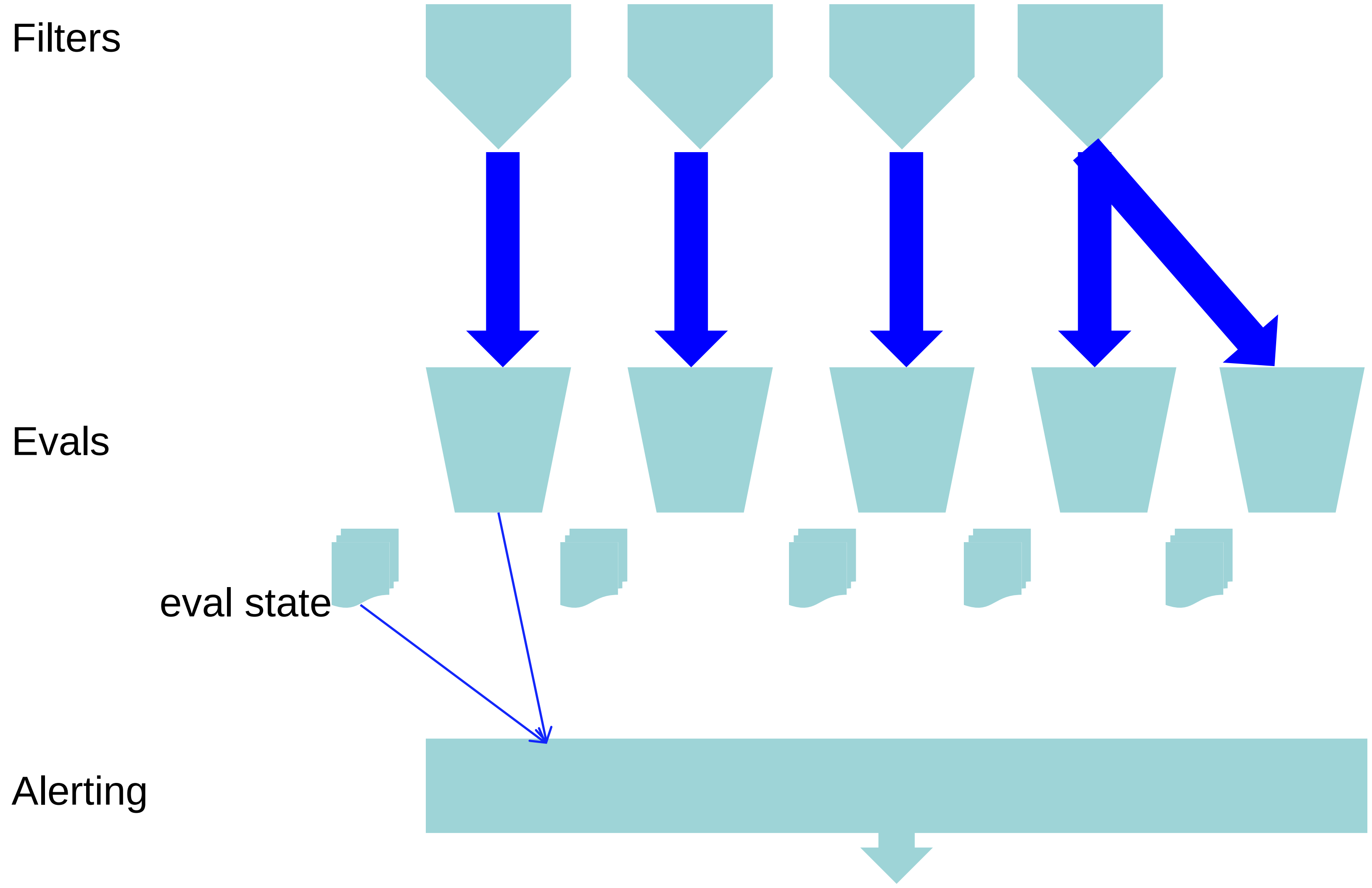
Filters and Evaluations

Each evaluation gets its flows from **one** filter

A filter can provide for multiple evaluations

A single filter is specified in the configuration file for each evaluation.

Connecting Filters -> Evals -> Alerts



Evaluations

The decision and analysis stage of pipeline

Majority of time and memory costs

Can have time restrictions:

- Alert if “this” happens in any 5 minute period

Made up of a number of independent checks

- E.g. Bytes > 1000 and packets > 500 in 5 minutes

Evaluations and Checks

Evaluations can be made up of multiple checks

- A check is where thresholds are specified
- Each check can be limited by its own time window
- Examples
  - Sum of Packets > 1000 in 10 minutes
  - Number of Unique Source IP Addresses > 10 in an hour
  - Total Flow Count > 10000 in 1 minute
- If all checks meet threshold, the evaluation alerts

Check Flow Processing

Each check is completely independent

- Pulls specific field value from flow
  - Ignores the rest of the flow record
- Aggregates that value with others from this file
- Timestamps aggregate and adds it to the list
- Updates state
  - Removes any aggregates that have timed out
  - Adds in the new aggregate from the current file
- Compares new state value against threshold

State Grouping

A check's state can be calculated for each unique value of the specified flow field

- We call it "for each"

Example: FOREACH SIP

- A different state value is stored and aggregated for each SIP found in the flow records
- Helps identify notable SIPs rather than saying that there might be an infected SIP in the network

Check Components

Type

- Method of collecting a state value

Threshold

- Value to compare to state value to check success

Operator

- The way to compare state value to threshold
- <, <=, >=, >, ==, !=

If {state value} {operator} {threshold} is true, the check returns success to the evaluation

Check Types

Total Count – Count number of flows received

- Ex: Count > 10000

Field Sum – Sum of the value of specified field

- Must provide the field name
- Ex: Sum PACKETS >= 500

Field Average – Average of the value of field

- Must provide the field name
- Ex: Average BYTES < 100

More Check Types

Unique Field Count - # Unique field values seen

- Need to declare field name
- Distinct DIP > 10
  - Success if more than 10 unique DIPs are seen

Proportion – How often a field value is seen

- Need to declare field name
- Need to declare field value
- Ex: Proportion PROTOCOL 6 > 75 PERCENT

Web Server Example

Identify web servers on the network

Analyze all traffic going out to port 80

Identifying features for a source address

- SIP sends more than 20,000 bytes in any 10 minute period
- SIP sends data to more than 10 different DIPs in that same 10 minute period

Web Server Example

Filter:

- dport == 80
- type == OUTWEB

Evaluation:

- FOREACH SIP
- Bytes > 20,000 bytes in 10 MINUTES
- Uniq DIPs > 10 in 10 MINUTES

Watchlist Evaluation

Check if the SIP or DIP is in the watchlist

- If so, alert on the flow record

Use evaluation type “EVERYTHING_PASSES”

- This alerts on all flow records

Filter:

- ANY_IP IN_LIST “watchlistFilename.set”

Evaluation:

- EVERYTHING_PASSES

Beacon Detection

Uses finite state beacon detection

- Outputs 4-tuple {SIP, DIP, DPORT, PROTOCOL}

Configurable parameters:

- Minimum number of beacons
- Minimum time window between beacons
- % variance on either side of established frequency

Sensor Outage

Presently the only file evaluation

Detects sensor outages

- Configuration contains list of sensors to inspect
- Reads sensor.conf to change names into IDs

Alerts if a flow file from a listed sensor does not arrive in the specified time window.

Internal Filters

Pipeline can build its own lists for filters

Same filtering capabilities of normal filters

They pull a specified field from each flow record that passes into a named list

These can be referenced by filters with IN_LIST

Internal filters are run before normal filters

IPv6 Tunneling

Use internal filtering

- Look for initial connection: DIP == ipv6 server addr
- Place that SIP in “IPv6 connectors” internal list

Second filter:

- SIP IN_LIST IPv6 connectors
- Proto == 41

Evaluation:

- Everything Passes

High Port Check

Goal is to identify passive traffic (ie. FTP)

- After port 21 traffic, transfers are on high ports

Uses an internal filter to look for flows with sport and dport > 1024

- Puts SIP and DIP into a list

If a port 21 connection is seen between the listed SIP and DIP, alert

- The port 21 flow will arrive after all of the high port flows as it stays open the entire time

Configurable Evaluation Features

Id

- A string used to uniquely identify an evaluation
- E.g. outgoing_watchlist_number_1

Eval type

- Another string used to group evaluations
- E.g. watchlist

Severity

- A severity value to be part of an alert triggered by pipeline for an eval

Output Type

- Result of evaluation: entire flow, SIP, FIVE_TUPLE, etc

List to send output – (non entire-flow evaluations)

- If evaluation isolates SIPs, they can be put into a list for use in other filters and evaluations, in addition to an alert

Alerting and Outputs

An evaluation that “alerts” creates an output

- Outputs contain:
  - Flow record
  - The FOREACH value (specified ip address in case of SIP)
  - Data values that caused the evaluation to alert
- They are placed in a list. Entries can time out.

At alert time, the valid outputs are packaged into alerts if the alert restrictions are met:

- X alerts in Y time or set to alert always

Alerts

When deemed able to alert, they contain:

- The flow record
- Evaluation name as identifier
- Metrics that triggered alert and its threshold
- Timestamp

Currently output to arcSight files

Can output to files and logs

Questions Contact

You can get the CERT NetSA tools from:

http://tools.netsa.cert.org

Questions on Pipeline or any of our tools:

netsa-help@cert.org