PNNL-SA-120800
PNNL-18428, Rev. 4

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

FLOWER Operations Guide

FLOWER Version 06 (flr06)

July 2017

DS Curtis

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RL01830
with Battelle Memorial Institute

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <http://www.ntis.gov/about/form.aspx>
Online ordering: http://www.ntis.gov

This document was printed on recycled paper.
(8/2010)

PNNL-SA-120800
PNNL-18428, Rev. 4

FLOWER Operations Guide

DS Curtis

July 2017

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RL01830
with Battelle Memorial Institute

Pacific Northwest National Laboratory
Richland, Washington 99352

Acknowledgments

 PNNL-SA-120800

Battelle IPID: 17042-E

Prime Contract No.: DE-AC06-76RL01830 with Battelle Memorial Institute

Acronyms and Abbreviations

FLOWER	FLOW analyzer
GCC	GNU Compiler Collection
pcap	packet capture files
PNNL	Pacific Northwest National Laboratory

Contents

1.0 Introduction

The FLOWER (FLOW analyzER) application is designed to start at boot time from RedHat or CentOS 7.x systemd as a daemon process to summarize network flows. FLOWER can also be used to read packets from a network interface or from packet capture (pcap) files that can be processed with the libpcap system library.

2.0 Runtime Options

2.1 Command Line Options

These options are only allowed on the command line and cannot be used in the configuration file. The format of each option below is short/terse command line option OR (`||`) long/verbose command line option. For example, you can enter `flower -h` or `flower --help` to display the help information.

`-h || --help`
Print a help message on how to use FLOWER and exits.

`-v || --version`
Print a version string including the Subversion revision number when the executable was produced, the name and version of the compiler used, the compiler optimization settings, whether debug symbols are in the executable or not, the version of the Boost.org C++ libraries used, the version of the libpcap library used, the date and time the executable was created, and the version of the *FLOWER Data Guide* showing what data this executable will produce. After printing this information, the program exits.

`-I || --interactive`
Force FLOWER to run in the foreground when running as the user `root`. The default is to run FLOWER as a daemon and log all output to syslog.

`-c <file> || --config-file=<file>`
Specify the configuration file to use when running FLOWER.

`-i <device> || --device=<device>`
Specify the network interface device to use when running FLOWER. The interface will be set to promiscuous mode.

`-f <file> || --input-file=<file>`
Specify the input file to read pcap data from when running FLOWER.

`-p <count> || --packets=<count>`
Exits after receiving `<count>` packets. Any pending records in the flow cache will be written to the output file and marked accordingly. **Note:** If count is `0`, FLOWER will read all the packets.

2.2 Command Line and Configuration File Options

These options can be used on the command line and in the configuration file. If used in both, the command line option will override the value specified in the configuration file. The format of each option below is short/terse command line option OR (||) long/verbose command line option OR (||) the format to use in the configuration file.

`-b <switch> || --buffer-packets=<switch> || buffer-packets=<switch>`
Capture packets in the packet buffer in case of unexpected termination: The default value of the switch is `0`. Set the switch to `1` to capture packets. **Note:** Only works with the `--use-ring` option.

`-m <count> || --max-packetbuffer-size=<count> || max-packetbuffer-size=<count>`
Maximum number of packets to keep in the packet buffer. The default value is `10000`. The value of count cannot be larger than 30000.

`-T <seconds> || --cache-timeout=<seconds> || cache-timeout=<seconds>`
Set the Session Inactivity Timeout (SIT) value (in seconds) to keep flows in the cache. The default value is `120 seconds`.

`-C <seconds> || --cache-forceout=<seconds> || cache-forceout=<seconds>`
Set the Session Force Out (SFO) value (in seconds) to force long running flows in the cache to be written to the data store. The default value is `900 seconds`.

`-S <seconds> || --summary-forceout=<seconds> || summary-forceout=<seconds>`
Set the data file rotation value (in seconds) to force existing data files to be closed and new data files to be created. The default value is `900 seconds`.

`-d <directory> || --output-data-dir=<directory> || output-data-dir=<directory>`
Specifies a data directory to write all output data files. The default value is `/data/flower`.

`-e <ext> || --output-file-ext=<ext> || output-file-ext=<ext>`
Specifies the output file extension to be used on all output data files. The default value is `dat`.

`-s <name> || --site-name=<name> || site-name=<name>`
Specified the site name to be used in the flow records and the output data files. There is no default value for the site name. It must be in the configuration file or specified on the command line.

`--max-flowcache-size=<number>`
The maximum number of simultaneous flows that can be held at the same time. This value also determines the amount of memory that is used by FLOWER when running. The value should be around 200,000 for moderately busy networks or running on a machine with limited memory resources. The value can be larger for busy networks as long as the system has sufficient memory resources.

`-r <switch> || --use-ring=<switch> || use-ring=<switch>`
Use the Linux kernel `PF_PACKET mmap` API rather than the libpcap API. The default value of the switch is `0`. Set the switch to `1` to use the `PF_PACKET mmap` API.

2.3 Configuration File Options

These options can only be used in the configuration file. The format of each option below is the format to use in the configuration file. Spaces can optionally be placed before and/or after the = to make the configuration file more readable.

`snaplen=<bytes>`
Limits the number of bytes placed into the kernel buffer to a maximum of `<length>` bytes rather than the default of `all bytes`. The default value is `65535`.

`skip-ipv4-packets=<switch>`
Skip IPv4 packets during processing. Note that IPv4 packets with a tunnel in them will not be skipped. All packets with tunnel information (IPv4 or IPv6) are always captured. The default value of the switch is `0`.

`output-file-group=<name>`

The group name (from `/etc/group`) to assign as the group owner of the output data files.

`suppress-ipv4-output=<switch>`

Suppress IPv4 output records. Note that all traffic is always processed, but if this switch is set to `1`, the IPv4 records are simply not written to the output file. The default value of the switch is `0`.

2.4 Examples

To create network flows from the `eth0` network interface:

```
flower --config-file=/etc/flower.conf --device=eth0
```

To create network flows from all the pcap files in the `/data/input` directory:

```
flower --config-file=/etc/flower.conf /data/input/*.pcap
```

To create network flows from the eth0 network interface and create output files in `/data/flower`:

```
flower --device=eth0 --output-data-dir=/data/flower
```

3.0 Runtime Environment

3.1 Device Lock Files

When FLOWER is started to monitor a network interface, it creates a lock file in `/tmp`. The name of the lock file is `flower.<interface>.lock` where <interface> is the name of the network interface that was specified on the command line or the configuration file. For example, if you run `flower -i eth0`, then there will be a lock file named `/tmp/flower.eth0.lock`. You can only run one instance of FLOWER on an interface. You can run multiple instances of FLOWER as long as each instance is monitoring a different interface.

The FLOWER application uses `fcntl` and advisory locks so that the lock files should be unlocked when the process dies. FLOWER makes every attempt possible to clean up and remove lock files when it exits. There are cases, like a power outage or forced reboot, that create a condition where the lock file does not get removed. If your system is configured so that `/tmp` is a `tmpfs` (memory), then the lock file will get removed on a reboot. If your `/tmp` is a real filesystem, you must take other measures to clean out `/tmp` on a reboot.

If you start FLOWER, and it exits with the message, "Shutting down: another instance of flower already has a lock on interface <interface>", you will have to remove the lock file manually.

3.2 Data Lock Files

All data files are created in the directory specified by the `output-data-dir` option on the command line or in the configuration file. The data file is created with a temporary filename that begins with a "`.`" character. When the file is closed by FLOWER, it is renamed to its final filename. Please refer to the *FLOWER Data Guide* for the actual filename conventions. For example, the temporary filename is `.20080924172430-pnnldev-flr06.dat`, and the final filename will be `20080924172430-pnnldev-flr06.dat`.

3.3 Data File Management

Since all the output files are written to the same output directory, you need to have a way to keep the output directory from accumulating too many files. This document assumes that software such as rsync is being used to perform data file management.

3.4 Logging/Output

If you run FLOWER as root, all output will go to syslog with a facility of daemon (`LOG_DAEMON`) and a priority of error (`LOG_ERR`). Please refer to the man pages for openlog(3) and syslog.conf(5) for more information. Using syslog allows the system administrator to create an entry in `syslog.conf` to direct the output of `daemon.err` to any desired location or use the default value. If you run FLOWER as a non-root user or interactively as root, all messages will go to the terminal.

After creating an output file, FLOWER prints out the statistics for the packets that were processed. The information includes the total number of packets read, the length of time of the processing, the number of packets processed per second (averaged over the time period), and the maximum number of simultaneous flows during the time period.

4.0 System Resources

The default parameters for FLOWER allows you to buffer up to 30,000 packets and handle 300,000 simultaneous flows while using less than 800 MB of RAM. Increasing the `max-flowcache-size` in the flower configuration file or command line option will also increase the memory required to run FLOWER.

When processing a data file, the output file(s) created are typically about 10% of the size of the pcap input file.

5.0 Output Messages

All messages consist of a caution, error, or fatal condition. All messages have the following format:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LEVEL
  File:      FILENAME.EXT (line:###)
  Function:  FUNCTION
  Message:   Error Code=CODE
  CONTEXT:  CONTEXT
     MESSAGE.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Where:

`LEVEL` is `CAUTION`, `ERROR`, or `FATAL`.

`FILENAME.EXT` is the name of the source code file where the message originated.

`###` is the line number in the `FILENAME.EXT` where the message originated.

`FUNCTION` is the name of the function where the message originated.

`CODE` is the condition associated with the message. For instance, the `CODE` can be `RangeError` if a value is outside of the expected range of values.

`CONTEXT` explains what the code was trying to do. For example, a context for creating a lock file would be "`Trying to create a lock file, /tmp/flower.eth0.lock`".

`MESSAGE` explains the condition. For example, "`Permission denied`".

5.1 CAUTION Messages

Caution messages are printed when FLOWER can still recover from an unexpected event. For example, a caution message is printed when an input file is missing or an input parameter is valid but outside the recommended range of values. Caution messages are typically related to parsing command line and configuration parameters. An example Caution message looks like the following:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
CAUTION
  File:      ProgramOptions.cpp (line:640)
  Function:  checkRange
  Message:   Error Code=RangeError
  Out of range error:  Exception
     The value of summary-forceout, 1000000, should be less than
     2500.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

5.2 ERROR Messages

Error messages are printed when FLOWER can still recover from an unexpected event but are more serious than a Caution message. For example, an Error message is printed when a lock file cannot be created. Error messages are typically related to parsing the data, creating data, or dealing with files. An example Error message looks like the following:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ERROR
  File:      OutputHelper.cpp (line:247)
  Function:  openLocked
  Message:   Error Code=FileIO
    File IO error:  Can't lock existing file
      Trying to open and lock file, /tmp/flower.eth1.lock (File
      exists)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

5.3 FATAL Messages

Fatal messages are printed when FLOWER cannot recover from an unexpected event. For example, a fatal message is printed when a system interrupt signal (`SIGINT`) is caught. Fatal messages are typically related to a missing resource like the configuration file or invalid command line or configuration file parameters. An example fatal message looks like the following:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
FATAL
  File:      ExceptionHandler.cpp (line:119)
  Function:  Handler
  Message:   Error Code=Signal
    Signal encountered:  Control-C or SIGINT
  Shutting down
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

6.0 Troubleshooting

If the flower process is not producing files but is still running, there are a few things you can do to find out what might be wrong. If the --`use-ring=1` command line or configuration file option is being used then files should always be produced even if there are no packets on the wire and the `suppress-ipv4-output=1` configuration file option is turned on.

Find out how many system resources `flower` is being used with the following commands:

```
ps --no-heading -C flower
lsof -p `ps --no-heading -C flower | awk '{print $1}'`
free –mt
grep flower /var/log/messages*
```

Find the raw packets that might have caused the problem by creating a pcap file in the output directory using the command:

```
kill -SIGSEGV $(ps --no-heading -C flower | awk '{print $1}')
```

7.0 Reporting Bugs

Please submit bug reports to `flower-support@pnnl.gov` and include as much information as possible to describe the problem.

Please include the output from the commands in the "Troubleshooting" section and the output from `flower -v` up to the copyright message. For example:

```
network packet FLOW analizER (flower)

flower version:           5.1.0
Compiled on:              Jun  9 2017, 14:38:43 (1497044321)
Compiled with:            gcc version 5.3.0 (GCC)
  Optimize Level:         -O3
  Debug:                  -g
  Boost library version:  /usr/local/boost
  pcap  library version:  libpcap version 1.5.3
  Data Guide version:       flr06
```

8.0 Ideas for Managing FLOWER Data

8.1 Use the FLOWER Data Guide

The FLOWER_Data_Guide_FLR06 document is very comprehensive and was used in the process of creating the sections that follow on how to work with FLOWER data. Please refer to the FLOWER_Data_Guide_FLR06 document to understand what the FLOWER data represents and how you want to manage FLOWER data as part of your overall data collection and management strategy.

The following sections are ideas on how to aggregate data and integrate it into Splunk or RDBMS. Please refer to the documentation for your specific version of Splunk or RDBMS to tailor these ideas to work in your environment.

WARNING: Be careful when using copy/paste operations from this document into your files. Some copy/paste operations between editors, operating systems, and terminal windows can mistakenly change a dash character (Unicode 2013) into a hyphen (Unicode 2014) which can be very difficult to troubleshoot. Similar problems can happen with quote, double-quote, and tick characters. When in doubt type the characters rather than depend on copy/paste operations.

8.2 Aggregate FLOWER Data

The easiest way to aggregate files from FLOWER sensors is to have them all push the data files they produce to a central server for further processing. If you do not remove files from the FLOWER sensor systems, the data files will eventually fill up the disk drives on the FLOWER sensor causing the system to fail. The command, `rsync`, is a good utility to automate the pushing of files to a central server and removing old files on each FLOWER sensor. To use rsync, you will need to create a pair of ssh keys and configure the FLOWER sensor to be able to connect to the central server using the ssh key.

The following is an example script to move files from a FLOWER sensor to a central server.

```
----------------------------------------------------------------------
#!/bin/bash

SSH_OPTIONS='ssh -i /path/to/ssh_public_key_file -T -o Compression=no -x'
SRC_PATH='/path/where/flower/files/are/generated'
DST_PATH='/path/to/put/flower/files'

rsync --remove-source-files -aHx --no-p --no-g --chmod=ug+rw -e "$SSH_OPTIONS" \
--numeric-ids $SRC_PATH username@central_server:$DST_PATH

exit $?
----------------------------------------------------------------------
```

8.3 Integrate FLOWER Data into Splunk

8.3.1 Configure Splunk Enterprise 6.X to Index FLOWER Data

Splunk needs to know that format of the data that it is going to index. Add the following section to your `$SPLUNK_HOME/etc/system/local/props.conf` file:

```
---------------------------------------------------------------------
[flower]
DATETIME_CONFIG =
FIELD_NAMES =
SOURCE,SITE,TIMET,DURATION,VLAN_ID,PROTO,FSSADDR_V4,FSDADDR_V4,FSSADDR
_V6,FSDADDR_V6,SRC_PAYLOAD,DST_PAYLOAD,SRC_BYTES,DST_BYTES,SRC_PACKETS
,DST_PACKETS,SRC_IP_OPTS,DST_IP_OPTS,SPORT,DPORT,SRC_ICMP_FLAGS,DST_IC
MP_FLAGS,ICMP_EARLY_LATE_FLG,SRC_TCP_OPTS,DST_TCP_OPTS,SRC_TCP_FLAGS,D
ST_TCP_FLAGS,TCP_EARLY_LATE_FLG,SRC_FIRST_TCP_TS,SRC_FIRST_TCP_SEQ,DST
_FIRST_TCP_TS,DST_FIRST_TCP_SEQ,SRC_LAST_TCP_TS,SRC_LAST_TCP_SEQ,DST_L
AST_TCP_TS,DST_LAST_TCP_SEQ,SRC_TCP_RETRANS,DST_TCP_RETRANS,TUNNEL_DEP
TH,TUNNEL_SADDR_V4,TUNNEL_DADDR_V4,TUNNEL_SADDR_V6,TUNNEL_DADDR_V6,TUN
NEL_PROTO,TUNNEL_SPORT,TUNNEL_DPORT,FRAGMENT_TYPE,IP_FRAG,ANOMALY
INDEXED_EXTRACTIONS = csv
KV_MODE = none
MAX_DAYS_AGO = 10500
NO_BINARY_CHECK = true
SHOULD_LINEMERGE = false
TIMESTAMP_FIELDS = TIMET
TIME_FORMAT = %s.%6N
category = Structured
description = FLOWER DATA FORMAT
disabled = false
pulldown_type = true
---------------------------------------------------------------------
```

8.3.2 Configure Splunk Enterprise 6.X to automatically index FLOWER Data

Add the following section to your `$SPLUNK_HOME/etc/system/local/inputs.conf` file:

```
---------------------------------------------------------------------
[monitor://path/to/flower/data_files
disabled = false
host = hostname
index = flower
sourcetype = flower
---------------------------------------------------------------------
```

You will need to change the value, `/path/to/flower/data_files`, in the `inputs.conf` file to the directory where you are storing files created by FLOWER. This directory would be where you are aggregating FLOWER data from all your sensors.

You will also need to change the value, `hostname`, in the `inputs.conf` file to be the hostname of the system running Splunk.

8.3.3 Example Splunk Enterprise 6.X Dashboard Panels to view FLOWER Data

Assuming you have created a Splunk dashboard named FLOWER, it will create a file named, $SPLUNK_HOME/etc/users/admin/search/local/data/ui/views/FLOWER.xml. You can replace the contents of the FLOWER.xml with the following to get a simple FLOWER dashboard that will show you a variety of information about your data:

```
-------------------------------------------------------------------------
<form>
  <label>FLOWER</label>
  <fieldset submitButton="false">
    <input type="time" token="field1">
      <label></label>
    </input>
  </fieldset>
  <row>
    <panel>
      <title>Traffic Volume over Time</title>
      <chart>
        <search>
          <query>index=flower extracted_SOURCE=0
| timechart  count</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option
name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone<
/option>
        <option
name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
        <option name="charting.axisTitleX.visibility">visible</option>
        <option name="charting.axisTitleY.visibility">visible</option>
        <option
name="charting.axisTitleY2.visibility">visible</option>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">linear</option>
        <option name="charting.axisY2.enabled">0</option>
        <option name="charting.axisY2.scale">inherit</option>
        <option name="charting.chart">line</option>
        <option name="charting.chart.bubbleMaximumSize">50</option>
        <option name="charting.chart.bubbleMinimumSize">10</option>
        <option name="charting.chart.bubbleSizeBy">area</option>
        <option name="charting.chart.nullValueMode">gaps</option>
        <option name="charting.chart.showDataLabels">none</option>
        <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
        <option name="charting.chart.stackMode">default</option>
```

```
        <option name="charting.chart.style">shiny</option>
        <option name="charting.drilldown">all</option>
        <option name="charting.layout.splitSeries">0</option>
        <option
name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
        <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
        <option name="charting.legend.placement">right</option>
      </chart>
    </panel>
  </row>
  <row>
    <panel>
      <title>Unique Source IPs</title>
      <single>
        <search>
          <query>index=flower extracted_SOURCE=0
| stats dc(FSSADDR_V4) as "Unique Source IPs"</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
      </single>
    </panel>
    <panel>
      <title>Unique Dest IPs</title>
      <single>
        <search>
          <query>index=flower extracted_SOURCE=0
| stats dc(FSDADDR_V4) as "Unique Dest IPs"</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
      </single>
    </panel>
    <panel>
      <title>Top 10 Dest Ports</title>
      <chart>
        <search>
          <query>index=flower extracted_SOURCE=0
| top DPORT</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="charting.chart">pie</option>
```

```
      </chart>
    </panel>
    <panel>
      <title>Top 10 Source Port</title>
      <chart>
        <search>
          <query>index=flower extracted_SOURCE=0
| top SPORT</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="charting.chart">pie</option>
      </chart>
    </panel>
  </row>
  <row>
    <panel>
      <title>Average Duration by Protocol</title>
      <chart>
        <search>
          <query>index=flower extracted_SOURCE=0
| stats avg(DURATION) by PROTO</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="charting.axisLabelsX.majorLabelStyle.rotation">-
90</option>
        <option name="charting.chart">column</option>
        <option name="charting.chart.stackMode">stacked</option>
      </chart>
    </panel>
    <panel>
      <title>Byte Ratio (Dest to Source)</title>
      <chart>
        <search>
          <query>index=flower extracted_SOURCE=0
| eval byte_ratio = DST_BYTES/SRC_BYTES
| stats avg(byte_ratio) by PROTO</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="charting.chart">column</option>
      </chart>
    </panel>
```

```
  </row>
  <row>
    <panel>
      <title>Top Source IPs</title>
      <chart>
        <search>
          <query>index=flower
| rex field=FSSADDR_V4
"(?&lt;first&gt;\d{1,3})(?&lt;second&gt;\d{3})(?&lt;third&gt;\d{3})(?&
lt;fourth&gt;\d{3})$"
| eval  first=tonumber(first), second=tonumber(second),
third=tonumber(third), fourth=tonumber(fourth)
| eval src_ip=first+"."+second+"."+third+"."+fourth
| search src_ip!=192.168.0.0/16
| timechart span=1h count by src_ip</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option
name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone<
/option>
        <option
name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
        <option name="charting.axisTitleX.visibility">visible</option>
        <option name="charting.axisTitleY.visibility">visible</option>
        <option
name="charting.axisTitleY2.visibility">visible</option>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">linear</option>
        <option name="charting.axisY2.enabled">0</option>
        <option name="charting.axisY2.scale">inherit</option>
        <option name="charting.chart">column</option>
        <option name="charting.chart.bubbleMaximumSize">50</option>
        <option name="charting.chart.bubbleMinimumSize">10</option>
        <option name="charting.chart.bubbleSizeBy">area</option>
        <option name="charting.chart.nullValueMode">gaps</option>
        <option name="charting.chart.showDataLabels">none</option>
        <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
        <option name="charting.chart.stackMode">stacked</option>
        <option name="charting.chart.style">shiny</option>
        <option name="charting.drilldown">all</option>
        <option name="charting.layout.splitSeries">0</option>
        <option
name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
```

```
        <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
        <option name="charting.legend.placement">right</option>
      </chart>
    </panel>
    <panel>
      <title>Top Destination IPs</title>
      <table>
        <search>
          <query>index=flower
| rex field=FSDADDR_V4
"(?&lt;first&gt;\d{1,3})(?&lt;second&gt;\d{3})(?&lt;third&gt;\d{3})(?&
lt;fourth&gt;\d{3})$"
| eval  first=tonumber(first), second=tonumber(second),
third=tonumber(third), fourth=tonumber(fourth)
| eval dest_ip=first+"."+second+"."+third+"."+fourth
| top dest_ip</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="count">20</option>
        <option name="dataOverlayMode">none</option>
        <option name="drilldown">cell</option>
        <option name="percentagesRow">false</option>
        <option name="rowNumbers">false</option>
        <option name="totalsRow">false</option>
        <option name="wrap">true</option>
      </table>
    </panel>
  </row>
  <row>
    <panel>
      <title>Top Unique Destination by Source</title>
      <chart>
        <search>
          <query>index=flower
| rex field=FSSADDR_V4
"(?&lt;first&gt;\d{1,3})(?&lt;second&gt;\d{3})(?&lt;third&gt;\d{3})(?&
lt;fourth&gt;\d{3})$"
| eval  first=tonumber(first), second=tonumber(second),
third=tonumber(third), fourth=tonumber(fourth)
| eval src_ip=first+"."+second+"."+third+"."+fourth
| search src_ip=192.168.0.0/16
| stats dc(FSDADDR_V4) as "Unique Dest" by src_ip
| sort - "Unique Dest"</query>
          <earliest>$field1.earliest$</earliest>
```

```
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option
name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone<
/option>
        <option
name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
        <option name="charting.axisTitleX.visibility">visible</option>
        <option name="charting.axisTitleY.visibility">visible</option>
        <option
name="charting.axisTitleY2.visibility">visible</option>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">linear</option>
        <option name="charting.axisY2.enabled">0</option>
        <option name="charting.axisY2.scale">inherit</option>
        <option name="charting.chart">column</option>
        <option name="charting.chart.bubbleMaximumSize">50</option>
        <option name="charting.chart.bubbleMinimumSize">10</option>
        <option name="charting.chart.bubbleSizeBy">area</option>
        <option name="charting.chart.nullValueMode">gaps</option>
        <option name="charting.chart.showDataLabels">none</option>
        <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
        <option name="charting.chart.stackMode">stacked</option>
        <option name="charting.chart.style">shiny</option>
        <option name="charting.drilldown">all</option>
        <option name="charting.layout.splitSeries">0</option>
        <option
name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
        <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
        <option name="charting.legend.placement">right</option>
      </chart>
    </panel>
  </row>
  <row>
    <panel>
      <title>Distinct Dest Ports by Protocol over IP</title>
      <chart>
        <search>
          <query>index=flower
| rex field=FSSADDR_V4
"(?&lt;first&gt;\d{1,3})(?&lt;second&gt;\d{3})(?&lt;third&gt;\d{3})(?&
lt;fourth&gt;\d{3})$"
| eval  first=tonumber(first), second=tonumber(second),
third=tonumber(third), fourth=tonumber(fourth)
```

```
| eval src_ip=first+"."+second+"."+third+"."+fourth
| search src_ip=192.168.0.0/16

| chart dc(DPORT) as "Unique Dest Port" over src_ip by PROTO</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option
name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone<
/option>
        <option
name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
        <option name="charting.axisTitleX.visibility">visible</option>
        <option name="charting.axisTitleY.visibility">visible</option>
        <option
name="charting.axisTitleY2.visibility">visible</option>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">linear</option>
        <option name="charting.axisY2.enabled">0</option>
        <option name="charting.axisY2.scale">inherit</option>
        <option name="charting.chart">column</option>
        <option name="charting.chart.bubbleMaximumSize">50</option>
        <option name="charting.chart.bubbleMinimumSize">10</option>
        <option name="charting.chart.bubbleSizeBy">area</option>
        <option name="charting.chart.nullValueMode">gaps</option>
        <option name="charting.chart.showDataLabels">none</option>
        <option
name="charting.chart.sliceCollapsingThreshold">0.01</option>
        <option name="charting.chart.stackMode">stacked</option>
        <option name="charting.chart.style">shiny</option>
        <option name="charting.drilldown">all</option>
        <option name="charting.layout.splitSeries">0</option>
        <option
name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
        <option
name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
        <option name="charting.legend.placement">right</option>
      </chart>
    </panel>
  </row>
</form>
--------------------------------------------------------------------------
```

8.4 Integrate FLOWER Data into an RDBMS

8.4.1 Create an RDBMS table for Holding FLOWER Data

Since there are so many RDBMS flavors such as MySQL, Postgres, and MS SQL SVR, the examples are generic and may need to be modified to work in your RDBMS in your environment.

The following is an example of some RDBMS DDL for creating a table in a database to hold FLOWER data:

```
-----------------------------------------------------------------------
CREATE TABLE Flower
(
   source                  smallint         NOT NULL,
   site                    varchar(20)      NOT NULL,
   timet                   numeric(27,6)    NOT NULL,
   duration                numeric(7,6)     NOT NULL,
   vlan_id                 smallint,
   proto                   smallint         NOT NULL,
   fssaddr_v4              varchar(15),
   fsdaddr_v4              varchar(15),
   fssaddr_v6              varchar(39),
   fsdaddr_v6              varchar(39),
   src_payload             bigint,
   dst_payload             bigint,
   src_bytes               bigint,
   dst_bytes               bigint,
   src_packets             bigint,
   dst_packets             bigint,
   src_ip_opts             bigint,
   dst_ip_opts             bigint,
   sport                   smallint,
   dport                   smallint,
   src_icmp_flags          varchar(4),
   dst_icmp_flags          varchar(4),
   icmp_early_late_flg     varchar(160),
   src_tcp_opts            bigint,
   dst_tcp_opts            bigint,
   src_tcp_flags           char(2),
   dst_tcp_flags           char(2),
   tcp_early_late_flg      varchar(96),
   src_first_tcp_ts        integer,
   src_first_tcp_seq       integer,
   dst_first_tcp_ts        integer,
   dst_first_tcp_seq       integer,
   src_last_tcp_ts         integer,
   src_last_tcp_seq        integer,
   dst_last_tcp_ts         integer,
```

```
   dst_last_tcp_seq        integer,
   src_tcp_retrans         bigint,
   dst_tcp_retrans         bigint,
   tunnel_depth            smallint,
   tunnel_saddr_v4         varchar(15),
   tunnel_daddr_v4         varchar(15),
   tunnel_saddr_v6         varchar(39),
   tunnel_daddr_v6         varchar(39),
   tunnel_proto            smallint,
   tunnel_sport            smallint,
   tunnel_dport            smallint,
   fragment_type           char(1),
   ip_frag                 smallint,
   anomaly                 bigint
);
-------------------------------------------------------------------------
```

You can create a good primary key with the following SQL DDL:

```
ALTER TABLE Flower
ADD CONSTRAINT PK_Flower PRIMARY KEY (timet, proto, sport, dport);
```

You can create a variety of SQL indices depending on how you want to query the data. The following SQL DDL is an index on the Protocol and Destination Port:

```
CREATE INDEX idx_proto_dport
ON Flower (proto, dport);
```

Other indices could be created on any combination of IPv4 addresses, IPv6 addresses, tunnel fields, and/or vlan_ids.

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

U.S. DEPARTMENT OF
ENERGY

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99352
1-888-375-PNNL (7665)

www.pnnl.gov