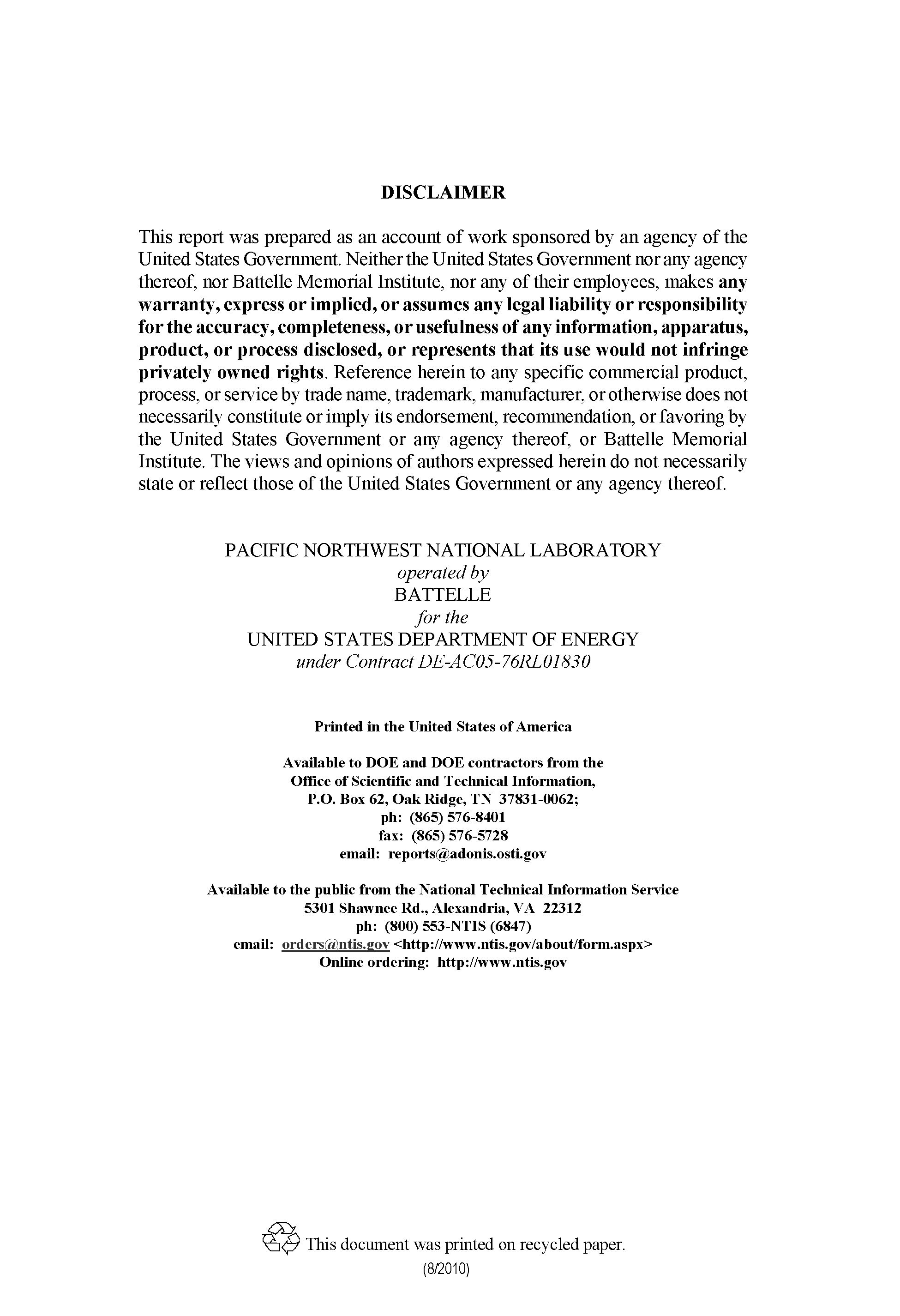
FLOWER Operations Guide

FLOWER Version 06 (flr06)

July 2017

DS Curtis



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

Copyright © (2011-2021) Battelle Memorial Institute. All Rights Reserved. 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

[Acknowledgments iii](#_Toc523925904)

[Acronyms and Abbreviations v](#_Toc523925905)

[1.0 Introduction 1.1](#_Toc523925906)

[2.0 Runtime Options 2.1](#_Toc523925907)

[2.1 Command Line Options 2.1](#_Toc523925908)

[2.2 Command Line and Configuration File Options 2.1](#_Toc523925909)

[2.3 Configuration File Options 2.2](#_Toc523925910)

[2.4 Examples 2.3](#_Toc523925911)

[3.0 Runtime Environment 3.1](#_Toc523925912)

[3.1 Device Lock Files 3.1](#_Toc523925913)

[3.2 Data Lock Files 3.1](#_Toc523925914)

[3.3 Data File Management 3.1](#_Toc523925915)

[3.4 Logging/Output 3.1](#_Toc523925916)

[4.0 System Resources 4.1](#_Toc523925917)

[5.0 Output Messages 5.1](#_Toc523925918)

[5.1 CAUTION Messages 5.1](#_Toc523925919)

[5.2 ERROR Messages 5.2](#_Toc523925920)

[5.3 FATAL Messages 5.2](#_Toc523925921)

[6.0 Troubleshooting 6.1](#_Toc523925922)

[7.0 Reporting Bugs 7.1](#_Toc523925923)

[8.0 Ideas for Managing FLOWER Data 8.2](#_Toc523925924)

[8.1 Use the FLOWER Data Guide 8.2](#_Toc523925925)

[8.2 Aggregate FLOWER Data 8.2](#_Toc523925926)

[8.3 Integrate FLOWER Data into Splunk 8.2](#_Toc523925927)

[8.3.1 Configure Splunk Enterprise 6.X to Index FLOWER Data 8.2](#_Toc523925928)

[8.3.2 Configure Splunk Enterprise 6.X to automatically index FLOWER Data 8.3](#_Toc523925929)

[8.3.3 Example Splunk Enterprise 6.X Dashboard Panels to view FLOWER Data 8.4](#_Toc523925930)

[8.4 Integrate FLOWER Data into an RDBMS 8.11](#_Toc523925931)

[8.4.1 Create an RDBMS table for Holding FLOWER Data 8.11](#_Toc523925932)

# 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.

# Runtime Options

## 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.

## 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.

-H <switch> || --header=<switch> || header=<switch>   
If the value of switch is 1, every output file will start with a CSV header row. The default value is 0. The value can only be 0 or 1.

-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.

## 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.

## 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

# Runtime Environment

## 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.

## 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.

## 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.

## 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.

# 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.

# 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”.

## 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.  
 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## 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)  
 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## 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  
 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# 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}')

# 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: 6.2.0

Compiled on: Feb 15 2019, 16:27:29 (2019-02-15T16:27:12)

Compiled with: g++-8

Version: 8.2.0

Optimize Level: -O2

Type: Release

Boost library version: 106800

pcap library version: libpcap version 1.8.1 -- Apple version 79.20.1

Data Guide version: flr06

Copyright (C) (2011-2021) Battelle Memorial Institute. All Rights Reserved.

# Ideas for Managing FLOWER Data

## 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.

## 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 $?

----------------------------------------------------------------------

## Integrate FLOWER Data into Splunk

### 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\_ICMP\_FLAGS,ICMP\_EARLY\_LATE\_FLG,SRC\_TCP\_OPTS,DST\_TCP\_OPTS,SRC\_TCP\_FLAGS,DST\_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\_LAST\_TCP\_TS,DST\_LAST\_TCP\_SEQ,SRC\_TCP\_RETRANS,DST\_TCP\_RETRANS,TUNNEL\_DEPTH,TUNNEL\_SADDR\_V4,TUNNEL\_DADDR\_V4,TUNNEL\_SADDR\_V6,TUNNEL\_DADDR\_V6,TUNNEL\_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

----------------------------------------------------------------------

### 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.

### 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>

----------------------------------------------------------------------

## Integrate FLOWER Data into an RDBMS

### 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 unsigned,

proto smallint unsigned NOT NULL,

fssaddr\_v4 varchar(15),

fsdaddr\_v4 varchar(15),

fssaddr\_v6 varchar(39),

fsdaddr\_v6 varchar(39),

src\_payload bigint unsigned,

dst\_payload bigint unsigned,

src\_bytes bigint unsigned,

dst\_bytes bigint unsigned,

src\_packets bigint unsigned,

dst\_packets bigint unsigned,

src\_ip\_opts bigint unsigned,

dst\_ip\_opts bigint unsigned,

sport smallint unsigned,

dport smallint unsigned,

src\_icmp\_flags varchar(4),

dst\_icmp\_flags varchar(4),

icmp\_early\_late\_flg varchar(160),

src\_tcp\_opts bigint unsigned,

dst\_tcp\_opts bigint unsigned,

src\_tcp\_flags char(2),

dst\_tcp\_flags char(2),

tcp\_early\_late\_flg varchar(96),

src\_first\_tcp\_ts integer unsigned,

src\_first\_tcp\_seq integer unsigned,

dst\_first\_tcp\_ts integer unsigned,

dst\_first\_tcp\_seq integer unsigned,

src\_last\_tcp\_ts integer unsigned,

src\_last\_tcp\_seq integer unsigned,

dst\_last\_tcp\_ts integer unsigned,

dst\_last\_tcp\_seq integer unsigned,

src\_tcp\_retrans bigint unsigned,

dst\_tcp\_retrans bigint unsigned,

tunnel\_depth smallint unsigned,

tunnel\_saddr\_v4 varchar(15),

tunnel\_daddr\_v4 varchar(15),

tunnel\_saddr\_v6 varchar(39),

tunnel\_daddr\_v6 varchar(39),

tunnel\_proto smallint unsigned,

tunnel\_sport smallint unsigned,

tunnel\_dport smallint unsigned,

fragment\_type char(1),

ip\_frag smallint unsigned,

anomaly bigint unsigned

);

----------------------------------------------------------------------

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.