User Manual
Version 1.4

Written by Gilberto Persico

Revision 1.4

Table of Contents

What is Fl0wer....6

How it works: Concepts....7

What you can do with Fl0wer....9

Application Identification & Classification....10

System Requirements....11

Software Requirements....11
Hardware Requirements....12
Hardware Sizing....13

Your 12 steps to have Fl0wer in production....14

Software Installation....15

Installing CentOS/RedHat 6 and 7....16
Installing Debian 7 and 8, Ubuntu, Devuan and other Debian derivatives....17
Uninstalling CentOS/RedHat 6 and 7....17
Uninstalling Debian 7 and 8, Ubuntu, Devuan and other Debian derivatives....18
The startup script file....18
Installing the license file....18
If something goes wrong....19

Signals to Fl0wer server....19

Data file rotation....19
Reset of data in memory....20
Scheduling daily rotation and cleanup....20

Configuration Directives....21

User and group....21
Logging....21
Listen Ports....22
Threading and Buffering....22
Datafiles....23
Fixups....24
Silent Decoding....25
Miscellaneous Features....25
Relationships....26
Flow Direction....26

NPAR.....27
IP Cache.....28
GeoIP.....29
Traffic Rules.....30
LUA Scripting.....31
Extra Functions available in the LUA interpreter.....34
TOR List.....36
Custom Network List.....36
DNS IPv4 and IPv6 Safe Server List.....37
NTP IPv4 and IPv6 Allowed Server List.....37
BGP IPv4 and IPv6 Peers Allowed List.....37
Top 100 Flows.....38
Security Last Flows.....38
Mail related parameters.....38

Network Security Features.....40

Risk Index.....40
Flow Deduplication.....40
Network Bogons.....41
IP Reputation.....41
Network Scans and Floods.....41
TOR Flows.....42
P2P Flows.....42
Unclassified Flows.....42
VPN Flows.....42
Authentication Flows.....43
Network Tunnel Flows.....43
Top 100 Flows.....43
External Management Flows.....43
Risk Flows.....44
Unknown DNS.....44
Unknown NTP.....44
Unknown BGP.....44
Long lived ICMP Flows.....44

Network Discovery Features.....44

RIP Routers.....45
OSPF Routers.....45
EIGRP Routers.....45
BGP Routers.....45
Netflow Learned Routers.....45
SNMP Clients.....46
SNMP Agents.....46
CIFS Servers.....46

NFS Servers....46
iSCSI Targets....46
FTP Servers....47
TFTP Servers....47
HTTP Servers....47
HTTPS Servers....47
Social Network Servers....47
SMTP Servers....48
SMTPS Servers....48
POP3 Servers....48
POP3S Servers....48
IMAP Servers....48
IMAPS Servers....49
Webmail Servers....49

Security Considerations....49

The Network side....49
The Host side....50

The Raw file format....50

DEMO Data Structure....51
COMPACT Data Structure....51
ANALYTIC Data Structure....52
FULL Data Structure....52

JSON API....56

General Functions....57
Network Lists functions....60
Flow Exporter functions....66
Flow functions....78
IP Cache Functions....90
Network Protocol functions....115
Reconfiguration functions....119
Top 100 Flows Functions....120
General Network Security Related Functions....121
Network Scans Related Functions....122
Network Floods Related Functions....122

The CLI utilities....123

Binary File Management....123
Daemon Configuration....123
User Management....124
Network Information & Reporting....124

The GUI Application (Fl0werUI)..128

Limitations..129

Support..130

Troubleshooting..130

Fl0wer Configuration File Example..133

The Performance Test Suite...139

External Software Licenses...140

OpenSSL and SSLeay License..141
libConfuse license...145
MaxMind GeoIP Legacy API License..146
libcurl License...147
Sqlite3 License..148
LUA Interpreter...149

What is Fl0wer

Fl0wer is a powerful Network Intelligence platform conceived to solve network visibility and network protection issues, to verify proper usage of company resources and to be used as a network blackbox to save all network traffic metadata, to track and solve problems that otherwise would be unnoticed. It runs on Linux (other Unixes like Solaris, AIX, NetBSD, etc. are available on demand), it is very scalable, has full support for IPv4 and IPv6 and it has an extremely low TCO.

While its engine (the server part) is a licensed product, all data, interfaces and JSON APIs are open-source, allowing for unmatched integrations and data exports to tools like Graphite or Apache Hadoop. It integrates seamlessly with Analytics tools like the ELK Stack (Elasticsearch, Logstash and Kibana) or Splunk. It uses the Netflow and IPFIX technologies to achieve its goals.

Fl0wer receives data flows from devices capable of exporting in one of the following formats:

- Netflow V1
- Netflow V5
- Netflow V9
- IPFIX (Netflow V10)

Fl0wer stores the received flows in a binary format that can be extremely compact, analytic or full, depending on your needs. The binary format is fully documented and can be quickly converted to whatever are your needs. The examples provided can be used to convert it into an SQLite (or Oracle) script that can be used to load data into a classic SQL database, or can be converted to a CSV (Comma Separated Value) text file that can be loaded into a spreadsheet for immediate analysis.

If needed, instead of saving in a binary format, Fl0wer can output directly JSON or CSV data in an immediate text-readable form that can be used to feed tools like Logstash for Elasticsearch™, Splunk™ or Apache Flume to load data into an HADOOP cluster.

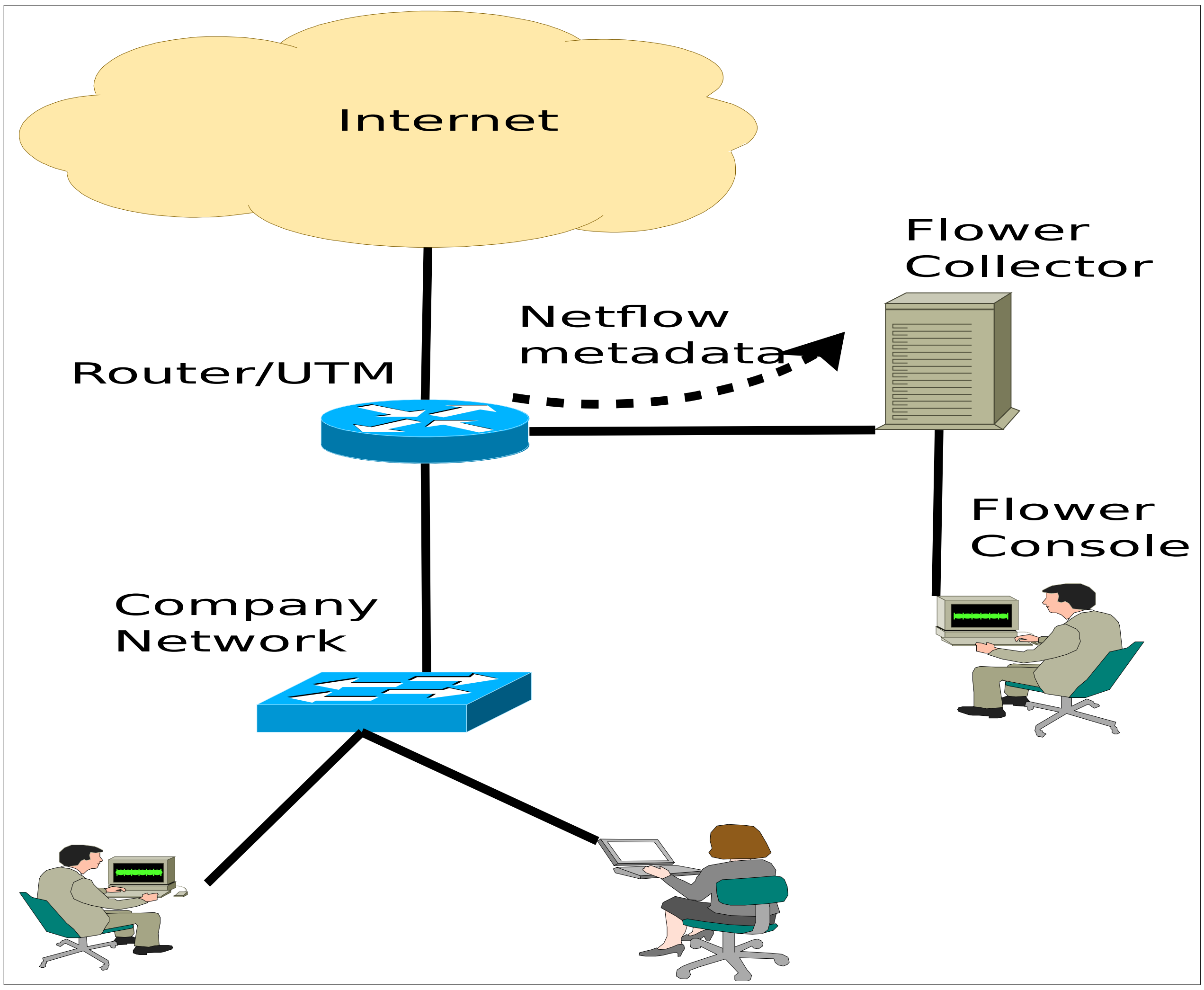
Depending on which are your needs, Fl0wer easily adapts to fit in them. If you just need an extremely high performance netflow collector for offline analysis, its scalability allows it to receive over 200K fps on a 2010 X64 hardware. If your requirements are to enrich the received data, you can easily configure several features like NPAR, Traffic Rules, several IP Lists to match data, GeoIP location of source and destination address (making use of the separately available GeoIP Enterprise database and GeoIP™ ISP Database from Maxmind™). You can even use it as a security tool since it integrates a LUA (http://www.lua.org) interpreter that dynamically executes custom-written code for every flow or template it receives. Obviously, the more features you enable, the lower the Netflow bandwidth it can sustain, choices are always a trade off between performance requirements and features enabled. Fl0wer is written in C language, makes use of multithreading and by means of the SO_REUSEPORT feature it can scale very well on systems with a lot of CPUs or cores, but cannot make miracles.

How it works: Concepts

Netflow/IPFIX are industry standards that summarizes the IP network traffic between two devices.

Explaining it in plain words, you can consider a Netflow/IPFIX stream as a phone call in a telephone bill. There is the IP address of the calling device, the IP address of the called device, the start time and end time of the conversation, the type of traffic, and the amount of bytes and packets transferred. The content of the conversation is not reported at all. In this way the protocol is very light and you can trace all the incoming and outgoing traffic from the device that exports the stream.

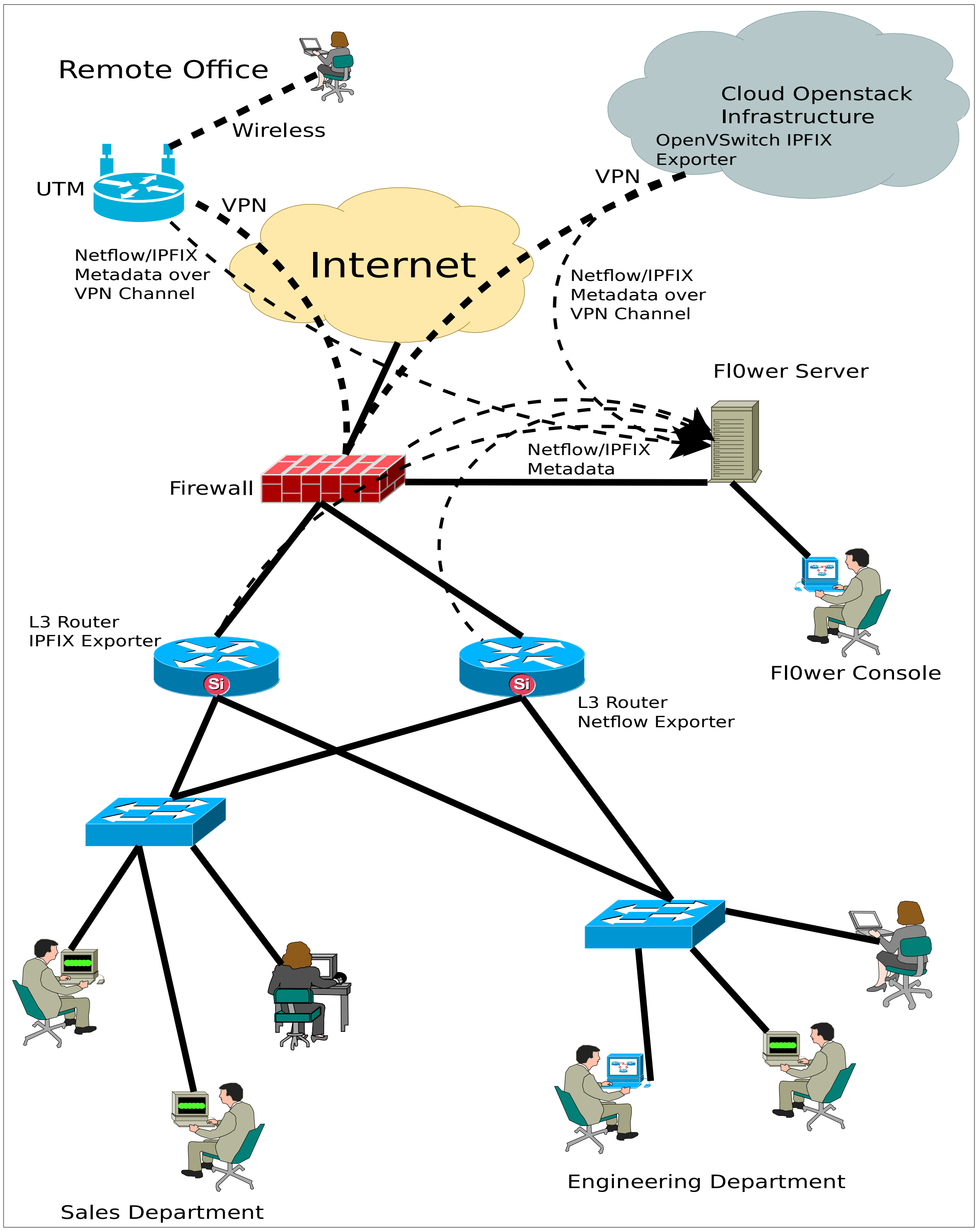
The following picture describes the basic concept behind Netflow/IPFIX:



As you can see in this simple example, the Fl0wer collector receives Netflow data from a Router or UTM device and stores it to disk, allowing information querying from a console using the provided tools.

In this example, only the traffic crossing the Router/UTM is tracked, all the communication on the "Company Network" switch is not tracked, since the Router/UTM device has visibility only of traffic "crossing" it or directed to it.

A more complex (and realistic schema) could be this one:


Remote Office
Wireless
UTM
VPN
Netflow/IPFIX Metadata over VPN Channel
Internet
Cloud Openstack Infrastructure
OpenVSwitch IPFIX Exporter
VPN
Netflow/IPFIX Metadata over VPN Channel
Fl0wer Server
Firewall
Netflow/IPFIX Metadata
L3 Router IPFIX Exporter
Fl0wer Console
L3 Router Netflow Exporter
Engineering Department
Sales Department

Fl0wer has no limits on the number of Router/UTM devices, and it is surprising to see how many vendors and software solutions export data in one of the supported Netflow/IPFIX formats.

As a quick and incomplete list you can get Netflow data from:

Routers/UTM: Mikrotik, Ubiquiti, Cisco, Huawei, Juniper, Sophos

Firewalls: CheckPoint GAIA, Palo Alto Networks, opnSense, pfSense, Linux ipt_NETFLOW.

Servers: openvswitch, softflowd on Linux, *BSD, etc.

Virtualization Platforms: VMWare ESXi >= 5 (with distributed vSwitch), Citrix (integrated openvswitch), Proxmox (using softflow or openvswitch), OpenStack Neutron (integrated openvswitch).

Fl0wer has been tested with most of them, revealing new and interesting perspectives for data analysis and security applications.

Conceptually, Fl0wer receives its input data (Netflow) on UDP port 2056 (it can be obviously changed) and listens for CLI-tools or GUI-Application requests on port 7443 (also this can be changed).

Incoming flows are instantly processed and decoded, submitted to its internal analyzer, then stored if required. The storage of flows is done using a separate thread for buffering, and when a buffer has to be stored, it is processed for all other things that are not done in real-time.

What you can do with Fl0wer

Fl0wer is a high performance solution that allows you to efficiently:

- receive and store network flows
- react and do things by means of user customized LUA Scripts
- analyze them, looking for TOR, P2P, Bad IP Reputation, VPNs, Tunnels, scans, etc.
- evaluate a Risk Index for each flow
- integrate easily with the most used Analytics software (like ELK or Splunk)

so, with the following features in mind, with an optimal installation of Fl0wer, between the other things, you can use it to:

- constantly monitor your network for anomalies
- know which kind of traffic travels on your networking infrastructure
- have a constantly updated, quick and immediate Risk Index of your whole network
- track resource usage
- use it as a network blackbox for forensic analysis
- using an Analytics software, you can easily drill to the level you want and integrate it with other security sources

In a few words: you can do Network Intelligence.

Application Identification & Classification

Rather than showing you port numbers, Fl0wer stores your traffic in memory in one of the following categories:

CATEGORY	**VALUE**	**TRAFFIC**
CATEGORY_UNCLASSIFIED	0	The rest of traffic.
CATEGORY_VPN	1	IPSec, OpenVPN, L2TP, etc.
CATEGORY_MAIL	2	SMTP, SMTPS, POP, IMAP, etc.
CATEGORY_WEB	3	HTTP, HTTPS, IRC, IRCS, GOPHER, NNTP, WHOIS
CATEGORY_MANAGEMENT	4	SSH, TELNET, SNMP, NUT, RSH, etc.
CATEGORY_DATA_STORAGE	5	NFS, SMB, iSCSI, FTP, etc.
CATEGORY_AUTHENTICATION	6	LDAP, LDAPS, YP, RADIUS, TACACS, KERBEROS, etc.
CATEGORY_NETWORK_OPERATION	7	DNS, NTP, OSPF, BGP, RIP, IS-IS, IGMP, ICMP, MPLS, SOCKS, etc.
CATEGORY_SYSTEMS_OPERATION	8	DHCP, BOOTP, RPC, NETBIOS, FINGER
CATEGORY_P2P	9	Bit Torrent, eMule, etc.
CATEGORY_VOICE_VIDEO	10	SIP, H.323, etc.
CATEGORY_PRINTING	11	CUPS, LPD, etc.
CATEGORY_NETWORK_TUNNEL	12	GRE, TOR, SOCKS
CATEGORY_DATABASE	13	MySQL, PostgreSQL, etc.
CATEGORY_SOCIAL	14	IRC, XMPP, FACEBOOK, TWITTER, etc.
CATEGORY_GAMING	15	Steam, Xbox Live, etc.
CATEGORY_UNWANTED	16	Malwares, Worms, Scans, Traffic with IP addresses with Bad Reputation or Bogons, etc.
CATEGORY_TRADING	17	Ethereum trading, etc.
CATEGORY_AUTOMATION	18	Scada, PLC, etc.

Obviously, in the binary file the traffic is stored with all the available information (depending on the format chosen), but the data in memory is classified using the above table. The above said categories will never change, and all of the details are available in the GUI-application under the "*Network Services→NPAR Mappings*" menu item when connected to a Fl0wer server. In future versions more categories could be added.

System Requirements

The Fl0wer platform is composed of three parts that are:

1. Fl0wer server daemon
2. CLI-tools
3. Fl0werUI GUI Application

In this release, the software runs only on Linux ***x64*** 64-bit platforms.

Software Requirements

The Fl0wer server daemon is tested and runs on the following Linux distributions:

- Devuan 1.0 64-bit
- Debian 7 and Debian 8 64-bit
- Ubuntu Desktop & Server 14.04 (using Debian 8 packages)
- Ubuntu Desktop & Server 16.04 (using Debian 8 packages)
- CentOS 6.7 and CentOS 7.3 64-bit
- RedHat 6.7 and RedHat 7.3 64-bit

►**Note:** On Centos and RedHat, the Fl0wer server will run even with "stock" base releases, but running the Fl0werUI client remotely (or on local display if running on a workstation) requires the above listed versions as a minimum. As a side note, having the last patches and updates installed is the best thing you can do on your servers. If you don't need to run the GUI application directly from the server, you can use the base versions.

There are also custom ports (available on demand) that are:

- Solaris 11 x64 and SPARC
- NetBSD x64 and NetBSD SPARC
- IBM AIX 6.1

More and different UNIX versions could be supported in the future, depending on business demand.

The open-source CLI-tools can be used on every platform supporting Python 3.4 (or greater) with the following Python modules (generally available via PIP):

- prettyprint

- simplejson
- tabulate
- pyOpenSSL

For your convenience, the open-source CLI-tools are pre-packaged with PyInstaller and ready to run on the same server platforms.

The open-source GUI-application can be used on every platform supporting Python 3.4 (or greater) with the following Python modules (generally available via PIP):

- tkinter and ttk
- matplotlib 2.0
- numpy
- pyOpenSSL
- simplejson

For your convenience, the open-source GUI application is pre-packaged with PyInstaller (py2app on OSX) and ready to run on the following platforms:

- Linux (CentOS/Redhat 6 and 7, Debian/Ubuntu 14.04 or later)
- Windows 32 (XP SP3 or greater) or Windows 64 (Windows 7 or greater) – Celeron M minimum processor required.
- Mac OSX 10.10 (Yosemite) or greater

►***For the prepackaged client versions, a display of at least 1280x800 is required as a bare minimum. A 1600x900 display is far more better and visible, and a full HD 1920x1080 display is optimal.***

Hardware Requirements

The Fl0wer server daemon scales up to your requirements so, the rule of thumb is: the more the FPS (Flows per Second) are requested, the more the CPU/RAM/Faster storage is needed. Regarding CPUs, a big L2 Cache improves performance dramatically.

As a minimum bare requirement, to use most of the features, you should have at least 1 Gbyte of RAM (if you decide to load in memory the Maxmind databases) but it is probably an extremely minimalistic bare minimum.

A more realistic and useful approach is to dedicate to it at least 4/8 Gbytes of RAM. Keep in mind that all statistic data, buffers, IP lists, IP Caches, internal structures, IP relations and all are stored in RAM for fast access and depending on your traffic patterns, they can grow up very fast- In a small ISP provider I have seen the IP Cache tree grow literally from 0 to over 1000000 entries in just a bunch of seconds, so you’d probably have to size it accordingly to this kind of parameters.

If you have a lot of Megabits of Netflow traffic, buffers help you to not loose a single packet, but they can take a lot of memory and disk writes have been seen in the range of 100/180 Mbytes per second. Losing packets means losing information that could be precious.

Take into account that to use the MULTI_THREADING feature, the kernel must support the SO_REUSEPORT feature as reported in https://lwn.net/Articles/542629/ . It is available in Linux Kernel versions >= 3.9 and it has been backported in RedHat/CentOS 2.6.32 kernel. There are no special requirements for the CLI-tools and the GUI-application (except what above said), but obviously, the faster the system, the better and the faster are the results. A 1600x900 pixels screen is strongly suggested.

Hardware Sizing

Some hardware performance number to get an idea (these have been measured using the Performance Suite included in the product) to size your system:

Platform	CPU	Cores	OS	Features	FPS	PPS
APU2C4 with 4 Gbytes RAM	AMD GX-412TC 1 Ghz	4	Devuan (debian8)	• Collector • NPAR • TorList • CustomNetList	> 8000	> 1000
Celsius R570-2 with 24Gbytes RAM	2x Intel Xeon 5670 2.93 Ghz	12	Ubuntu 14.04	• Collector • NPAR • TorList • CustomNetList	> 150000	> 35000

FPS = Flows Per Second

PPS = Packets Per Second

It is clear that the more the FPS and the PPS, the more powerful the CPU needs to be. Also, you have to carefully enable or disable the features you need. Probably, if you need to capture a lot of flows per second, post-processing them is the best approach, but if your FPS rate is low, maybe near-realtime processing features can be more helpful. To help to make an estimate, a Gigabit Link to Internet, full at 50% of usage (ie ~ 500Mbit/s) produces an average of 800 FPS.

Regarding the Storage, it depends on the level detail you want to have.

Compact storage model is a very simple approach, since it stores only the basic Netflow data without enrichment (it sums to 216 bytes per flow).

Analytic adds NPAR to Compact, staying at about 480 bytes per record, it’s the default.

Full stores about 3360 bytes per flow and should be used only in very particular cases when you need to store everything or want to experiment things.

Considering the above with a 500Mbit of average traffic at 800 FPS, the math follows:

Format	FPS	Hourly	Daily (24 h)	Monthly (30 d)
Compact (216 bytes)	800	593 Mbytes	~14 Gbytes	420 Gbytes
Analytic (472 bytes)	800	~ 1.3 GBytes	~31.2 Gbytes	~936 GBytes
Full (3600 bytes)	800	~ 9.8 GBytes	~ 235 Gbytes	7050 GBytes

Text formats like CSV or JSON are even larger. But if you rotate daily these files, data compression with gzip or bzip2 can save at least 50% of used space. Further, using JSON and CSV or CSVFULL you can easily integrate Netflow/IPFIX data into Analytics solutions like Elasticsearch or Splunk, as example.

Your 12 steps to have Fl0wer in production

Fl0wer comes with several predefined configurations. Here are the steps to have it quickly up & running.

1. Write down your internal network topology and configure your flow exporters. There are tons of guides on the Internet if user manuals are not clear enough. In doubt, ask, I probably can help.
2. Choose your x64 hardware, install a supported Linux distribution (Debian/Ubuntu or CentOS/RedHat) and configure networking properly. Check for default firewall rules, UDP 2056 is required to receive data from the exporters and TCP/7443 is used for the API, FlowerUI client and CLI tools.
3. Install the Fl0wer package.
4. Go in the */opt/fl0wer/etc* and setup your configuration in *fl0wer.conf*.
5. Review & Adjust the parameters that fits your requirements.
6. Install (if available) the relative MaxMind databases as per instructions in this manual. Be sure that the fl0wer user can read them !
7. Setup the startup script in /etc/init.d using the provided */opt/fl0wer/scripts/fl0wer-startup-script* as example.
8. Setup the data rotation crontab.
9. Start the Fl0wer Daemon. Predefined defaults should work in most cases, but you can adjust everything you want.
10. Install the GUI Application on your Linux/Windows/Mac.
11. Connect to the Fl0wer server with the GUI Application.
12. Enjoy and start checking your network !

Software Installation

The Fl0wer platform is packaged using native Linux or UNIX distribution package mechanism, so the Debian and Devuan version can be installed like any other .deb package, the CentOS/RedHat version is a traditional .rpm package.

The fixed installation path is in /opt/fl0wer and it creates a tree like this:

Path	Use
/opt/fl0wer	The root tree of the package. Cannot be changed.
/opt/fl0wer/bin	This directory contains the binary CLI tools,the GUI Application and the supportinfo script.
/opt/fl0wer/etc	This directory contains the configuration file, the user database, the license file, the certificate files for the TLSv12/SSL server part that works with the CLI and the GUI Application.
/opt/fl0wer/data	This directory is where Fl0wer stores its binary data by default. Can be changed in the configuration file.
/opt/fl0wer/geo	This directory is where Fl0wer searches for the MaxMind GeoIP data files.
/opt/fl0wer/logs	This directory is where Fl0wer stores its log files by default. Can be changed in the configuration file.
/opt/fl0wer/iplist	This directory is where Fl0wer stores its log files by default. Can be changed in the configuration file.
/opt/fl0wer/luascripts	This directory is where Fl0wer looks for LUA scripts if enabled. Can be changed in the configuration file.
/opt/fl0wer/sbin	This directory contains the Fl0wer daemon.
/opt/fl0wer/scripts	This directory contains example script to startup the Fl0wer daemon and for data rotation.

By default, it runs as userid: *fl0wer* and groupid: *fl0wer*, although this can be changed (even if not suggested). The server daemon does not need *root* privileges since by default it binds on a > 1024 UDP port.

Before installation, it is suggested to tune the following parameters in */etc/sysctl.conf*:

```
net.ipv4.ip_local_port_range = 9000 65000
net.core.rmem_default = 16777216
net.core.rmem_max = 33554432
net.core.wmem_default = 16777216
net.core.wmem_max = 16777216
net.ipv4.udp_mem = 16777216 134217728 268435456
fs.aio-max-nr = 1048576
fs.file-max = 6815744
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_max_tw_buckets = 400000
net.ipv4.tcp_max_orphans = 60000
net.core.somaxconn = 40000
net.ipv4.tcp_synack_retries = 3
net.ipv4.tcp_fin_timeout = 5
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_max_syn_backlog=40000
net.core.netdev_max_backlog=10000
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_moderate_rcvbuf = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
```

The suggested parameters should be tuned according to your server requirements referring to the Linux documentation about them. This is what is used on an HP N54L server with 1 AMD Turion II CPU with 4Gbytes of RAM and 9Tbytes of RAID5 storage for development. It runs CentOS 6.7 and it handles 16 IPv4 assorted Flow Exporters (Mikrotik, Ubiquiti, Cisco, openvswitch, softflowd), 1 IPv6 Flow Exporter (using softflowd) with Fl0werd averaging at 3% of total CPU usage.

The following parameters have to be tuned according to your server requirements in */etc/security/limits.conf*:

```
*                -       nofile          65535
*                -       stack           65536
*                -       memlock         unlimited
*                hard    rtprio          99
*                soft    rtprio          99
```

For some operations, Fl0wer makes a heavy use of the stack (recursive operations has a price !) so a bare minimum of 64Mbytes of stack available per process is required to run. Also, Fl0wer tries to run with the highest priority available so it won’t loose any single packet of data.

►**Important:** if you are getting errors in pthread_create in the error log, you probably did not set correctly the rtprio parameters !

►**Important:** if the Fl0wer daemon is running but you are not receiving flows or cannot connect to the JSON Interface, double check your iptables settings on the server !

Installing CentOS/RedHat 6 and 7

Once you downloaded the package, you have a .tgz file, which is really a compressed tar file.

Simply uncompress it with a command like:

```
$ tar zxvf fl0wer-version.tgz
```

then become root, go in the directory where you extracted the compressed tar file and execute the command:

```
# rpm -Uvh fl0wer-version.rpm
```

if you want to install also the CLI-Tools sources, GUI Application sources and headers, install the fl0wer-devel-version.rpm package too.

►Note: if the /opt/fl0wer/bin/Fl0werUI fails to run on a brand new installed Red Hat Enterprise 7.0 with the following error:

```
$ /opt/fl0wer/bin/Fl0werUI
Traceback (most recent call last):
  File "fl0werui.py", line 22, in <module>
  File "/usr/lib/python3.4/site-packages/PyInstaller/loader/pyimod03_importers.py", line 389, in load_module
  File "tkinter/__init__.py", line 38, in <module>
  File "/usr/lib/python3.4/site-packages/PyInstaller/loader/pyimod03_importers.py", line 573, in load_module
ImportError: /tmp/_MEIdk90L9/libX11.so.6: undefined symbol: xcb_poll_for_reply64
Failed to execute script fl0werui
```

Then you simply need to install the *libxcb* package with the following command:

# yum install libxcb

Note: Although not recommended, if you want to run the Fl0werUI client on a server installation of RHEL/Centos 6 you should install the following packages:

```
# sudo yum groupinstall "Remote Desktop Clients"
# sudo yum groupinstall "X Window System"
# sudo yum install xorg-x11-server-utils
# sudo yum install xorg-x11-server-common
# sudo yum install xorg-x11-apps
# sudo yum install libX11
# sudo yum install libX11-common
# sudo yum install xorg-x11-fonts-75dpi xorg-x11-server-common
# sudo yum install libSM libXt libICE
# sudo yum install xorg-x11-fonts-ISO8859-1-100dpi
# sudo yum install gtk2
# sudo yum install mesa-libGL libXcomposite
# sudo yum install cairo dmidecode
# sudo yum install hal
# sudo yum install libX11
# sudo yum install libpng libjpeg
# sudo yum install giflib
# sudo yum install freetype
# sudo yum install xorg-x11-fonts-misc urw-fonts xorg-x11-fonts-ISO8859-15-75dpi xorg-x11-fonts-Type1
```

Installing Debian 7 and 8, Ubuntu, Devuan and other Debian derivatives

Once you downloaded the package, you have a .tgz file, which is really a compressed tar file.

Simply uncompress it with a command like:

```
$ tar zxvf fl0wer-version.tgz
```

then become root, go in the directory where you extracted the compressed tar file and execute the command:

```
# dpkg -i fl0wer-version.deb
```

if you want to install also the CLI-Tools sources, GUI Application sources and headers, install the fl0wer-devel-version.deb package too.

Uninstalling CentOS/RedHat 6 and 7

To uninstall the package, first stop the Fl0wer daemon with a command like:

```
$ /opt/fl0wer/scripts/fl0wer-startup-script stop
```

then, as root, issue the command:

```
# rpm –erase fl0wer flower-devel
```

The fl0wer user and group will remain, they have to be deleted manually. If you are upgrading from an evaluation version to a licensed version don't remove anything and simply proceed with the installation of the licensed package and copy of the license file.

Uninstalling Debian 7 and 8, Ubuntu, Devuan and other Debian derivatives

To uninstall the package, first stop the Fl0wer daemon with a command like:

```
$ /opt/fl0wer/scripts/fl0wer-startup-script stop
```

then, as root, issue the command:

```
# dpkg --removefl0wer flower-devel
```

The fl0wer user and group will remain, they have to be deleted manually. If you are upgrading from an evaluation version to a licensed version don't remove anything and simply proceed with the installation of the licensed package and copy of the license file.

The startup script file

You can find an example startup script in the path */opt/fl0wer/etc/fl0wer-startup-script* . You can customize it but it should work pretty well on most SysV init UNIX distributions. Usually, a command like:

# ln -s */opt/fl0wer/scripts/fl0wer-startup-script /etc/rc2.d/S99fl0wer*

will set it up to start at system boot.

► If you are using a *systemd* based distribution, you are on your own, I don't support it.

Installing the license file

The Fl0wer server does not work without a license file. The store will send you an automatic mail containing an activation key that is used as a proof to claim the license file in case of problems.

The license file is sent to the "Licensee Mail" specified address within 24/48hours after the order approval on the https://fl0wer.me online store. The license file is contained within an encrypted zip file, that is encrypted using the activation key you received with your order confirmation as the password.

►**Never share the activation key and don't use it as the activation file, it won't work.**

The license file sent must be copied in /opt/fl0wer/etc/license.lic file and the entry:

```
license_file = /opt/fl0wer/etc/license.lic
```

has to be added or adjusted in the */opt/fl0wer/etc/fl0wer.conf* file.

The license does not prevent the copy of the software, but stores in an encrypted way the user data to which the license is issued.

►**It is complete user responsibility to not share the license file and not distribute nor the package (in any part) nor the license file to anyone else, and completely respect the Right To Use License Agreement signed when the software is bought. Read it carefully since no refunds will be provided.**

If something goes wrong

The preinstall and postinstall scripts do the following things:

Preinstall

1. create the fl0wer group
2. create the fl0wer user (within the fl0wer group) with /opt/fl0wer as the home directory
3. save the configuration files (and the files in the iplist directory if any)

Install

- unpack the files in the packages

Postinstall

1. create a certificate file (in the licensed version) – if it already exists, this step is skipped
2. set the correct owner permissions (400) for the /opt/fl0wer/etc/certificate* files
3. set the correct owner permissions (400) for the /opt/fl0wer/etc/fl0werusers.db

Signals to Fl0wer server

Data file rotation

The data file is rotated whenever the Fl0wer daemon receives a USR1 signal. When the USR1 signal is received, the Fl0wer daemon flushes its data in the buffers, creates a new binary data file and moves the data file to the same path but with the *.prev* extension.

As example, if you want to have the history of 1 day, you can simply schedule a crontab that at midnight executes a command line like:

/bin/kill -USR1 `/sbin/pidof fl0werd` 2>&1 > /dev/null

So, the *datafile.prev* would contain the data of the previous day and the *data file* is the current data.

If you want to keep the history for more, a sample script to execute in crontab could do something like:

```
#!/bin/sh
```

```
/bin/mv /data/netflow/netflowdata.flw.prev /data/netflow/netflowdata.flw.`date +%Y%m%d%H%M` 2>&1 > /dev/null
```

giving the *datafile.prev* a new name like *datafile.prev.31122016235900*. This script is available in the path */opt/fl0wer/scripts/rotatedata.sh* .

Reset of data in memory

Fl0wer can be instructed to cleanup its data in memory (only in RAM, the binary file is not touched) including all counters.

Sending a signal USR2 does the job, there is also an API that allows you to do this remotely.

Scheduling daily rotation and cleanup

This is a pretty easy task easily accomplished using the cron UNIX facility. Setup correctly your data rotation script (you can use the pre-provided one in /opt/fl0wer/scripts/rotatedata.sh).

Setup crontab entries for the fl0wer user like this using the *crontab -e* command:

```
0 0 * * * /bin/kill -USR1 `/sbin/pidof fl0werd` 2>&1 > /dev/null

1 0 * * * /bin/kill -USR2 `/sbin/pidof fl0werd` 2>&1 > /dev/null

2 0 * * * /opt/fl0wer/scripts/rotatelog.sh 2>&1 > /dev/null
```

This will rotate your data file at midnight, will cleanup all data counters in memory and will rename the old file with a timestamped name.

Obviously, using this mechanism, you can setup whatever policy you prefer.

Configuration Directives

Most configuration directives are stored in the */opt/fl0wer/etc/fl0wer.conf* file and these are read at start-up of the server daemon. These directives change the configuration and behavior of the Fl0wer daemon. In this release, they cannot be changed at run time, probably in future versions some support will be added.

►A rule of thumb that should never be forgot is that in all cases, **the first matching entry** (for a subnet, a rule or whatever) is the one that applies. For example, if you first describe a big network like a /16, then describe smaller networks that are *inside* the above said /16, they won't be considered and the /16 network will be used. Also, **the first matching rule** is the one that applies.

User and group

The Fl0wer daemon needs to run as a specific unprivileged user and group. By default, at software installation, the fl0wer group and users are created and in the configuration files the following values are set:

```
user = fl0wer
group = fl0wer
```

Although using different users can work, free support will not be available for different configurations.

Logging

By default, Fl0wer writes logs in the following locations:

```
applog = /opt/fl0wer/fl0wer.log
errlog = /opt/fl0wer/fl0wer_error.log
perflog = /opt/fl0wer/fl0wer_perf.log
```

The applog is the normal application log; there are written normal usage messages, like user access, start up,etc.

The errlog is the error log; there are written error messages.

The perflog is where Fl0wer stores its performance data, that can be graphed using GNUplot or other tools after a bit of processing. It is enabled using the flag:

```
enable_perflog = no
```

The perflog flag should NEVER be enabled, unless instructed by support, due to performance degradation it can introduce.

If you want to send to syslog the messages from Fl0wer you simply enable it using the configuration file parameter:

```
enable_syslog = no|yes
```

By default, it is disabled.

Listen Ports

Fl0wer normally listens on two service ports:

- one UDP port for incoming Netflow packets (by default it uses UDP/2056) and is set using the configuration file parameter:

```
netflow_port = 2056
```

- one TCP port for incoming client requests (as example, for the GUI or flcli-* command line tools) and by default it uses the TCP/7443. It is set in the configuration file using the parameter:

```
clientport=7443
```

► Remember that if you are using a host based firewall on the Fl0wer server, you will need to open the above said ports.

If you have Netflow exporters using IPv6 to export their flows, you should properly configure your IPv6 network stack and enable the configuration file parameter:

```
enable_listen_ipv6 = yes|no
```

This parameter is disabled by default and has the value of “no”.

The dialog on the TCP/7443 port with the GUI and the CLI tools is done using by default a TLSv1.2 channel; if you are building things and/or using the client on very old operating system versions not supporting TLSv1.2, you can enable SSLv2 using the following configuration parameter:

```
disable_tls=yes
```

Threading and Buffering

The Fl0wer server normally runs using several threads to capture and process the incoming Netflow data. The *maxthreads* parameter specifies how many threads the Fl0wer server *should* use. This number should be equal to the number of virtual CPUs available on the platform (a quick and dirty way to find it is to use a command like: *cat /proc/cpuinfo | grep processor | wc -l*). If the number of virtual CPUs corresponds to *maxthreads*, the requested threads are started and linked to the relative CPU. More threads can be run, but if the software saturates the virtual CPUs, it is pretty useless.

As example, if you are running on a single Intel X5670 CPU, this one has 6 cores, with Hyper Threading enabled it maxes out 12 “virtual CPUs”. Running more than 12 processing threads it is a waste of resources.

The default values is:

```
maxthreads = 2
```

Buffering is a key part of a high-performance product and so is Fl0wer. The *maxnetbuf* parameter sets the number of buffers where the kernel stores the UDP packets before dropping them. This number is expressed in Megabytes and has to be multiplied for the number of running *maxthreads*.

The default value is:

```
maxnetbuf=8
```

The Fl0wer server storage subsystem uses a double-buffering mechanism to optimize the disk-writes, allowing it to reach very high performance that would otherwise be impossible.

The *buffers* parameter specifies how many flows should be stored in memory before triggering a disk write thread and should be multiplied by 2 and then by the number of running threads. A single flow memory record is about 3KBytes of information while decoded and enriched. As you can easily guess, it is an important parameter to be used when tuning for performance.

The default value is:

buffers = 1000

The *flows_to_keep_in_ram* parameter sets the cumulative number of flows that can be downloaded when using the *GetFlowsAll*, *GetTCPv4Flows*, *GetUDPv4Flows*, *GetTCPv6Flows*, *GetUDPv6Flows* and *GetOtherFlows* download functions. These are used both in the Fl0werUI GUI Application and in the flcli-rtgetflow* commands.

The default value is:

```
flows_to_keep_in_ram = 2000
```

The *thread_nst_buffer* is a special parameter that is used when computing the correct *flows_to_keep_in_ram* value by the Fl0wer daemon when running. The rule of thumb is that *flows_to_keep_in_ram* has to be a number that can be divided (without remainder) by the *thread_nst_buffer* parameter. The default value should be ok for most cases. The concept is pretty complex to explain and is related to how many buffers are copied in memory from the *buffers* to the NST (Network State Table) when flows are decoded and should be ready to be sent to the requesting client. Simply converting each flow one by one doesn’t provide with the necessary speed, so *memmove()* operations are preferred since they are extremely fast.

The default value is:

```
thread_nst_buffers = 10
```

Datafiles

The parameter netflow_datafile informs the Fl0wer server where the binary file (or JSON or CSV data) have to be stored in the file system.

The default value is:

```
netflow_datafile = "/opt/fl0wer/data/netflowdata.flw"
```

The parameter *storage_format* informs the Fl0wer server how the information should be stored.

It can be one of the following:

- demo   (16600 bytes record size)
- compact (216 bytes record size), available only in the licensed version.
- analytic (480 bytes record size), available only in the licensed version.
- full (3360 bytes record size), available only in the licensed version.
- csv (variable length), available only in the licensed version.
- csvfull (variable length)
- json (variable length), available only in the licensed version.
- none (variable length), available only in the licensed version.

The default value is:

storage_format = csvfull

Note: If required, a single line JSON data format can be used to feed data to systems like Apache Flume or Logstash in a way similar to LOG4J. You will need to add the feature:

json_singleline = yes

to the configuration file.

Note: csvfull is the default file format for Fl0wer versions 1.3 onwards, to ease software integration with Analytics softwares like ELK or Splunk and is available in both evaluation and commercial version, even though with only one thread decoding & analyzing flows, packet loss is possible.

Fixups

Some Flow Exporters, due to bugs or using different standards, exports the timestamps of the start & stop of the flow in a way different from IETF IPFIX or Official Netflow documentation data. Fl0wer can checkup and fix the wrong timestamps using the *fixup_timestamp* parameter. It will use the time when the flow packet is received instead.

The default value is:

```
fixup_timestamp=yes
```

During the development of Fl0wer daemon, it has been noticed that some host-based exporters like ipt_NETFLOW or softflowd also export the internal flows between some services of the same host, using 127.0.0.1 as source and destination. Fl0wer can fixup this replacing the 127.0.0.1 IP address with the IP Address of the Flow Exporter, which makes more sense when analyzing data.

The default value is:

```
fixup_localhost=yes
```

The following parameters allows the user to set the fl0wer ipaddress and enable or disable tracking of its own DNS queries in the flows.

Values can be set as the following examples:

```
fl0wer_ipaddress = "1.1.1.1" (or your public IP address)
track_fl0wer_dnsqueries = 0
```

track_fl0wer_dnsqueries values can be 0 (disabled) or 1 (enabled).

Silent Decoding

Fl0wer normally does not writes too much information in its logfiles, but if required (or requested by support), some more detail about the decoding process (and more) can be seen. The default value is:

```
silent_decoding = no
```

Miscellaneous Features

Fl0wer can obviously be configured to use a DNS to resolve IP addresses it sees during netflow collection. If incoming Netflow traffic is too high, it is strongly suggested to disable this feature, packet loss can be faced if name resolution is too slow.

The default value is:

```
enable_dns = yes
```

Also, Fl0wer has been tested with OpenVSwitch that has the feature to export also some Layer 2 Flows. This feature can be enabled or disabled using the *enable_l2* configuration file parameter.

The default value is:

```
enable_l2 = yes
```

The licensed version allows the user to use a custom generated certificate for TLSv1.2 encryption of data for the JSON API on TCP port 7443, by means of the following directives:

```
pem_cert_datafile = /opt/fl0wer/etc/certificate.pem
pem_key_datafile = /opt/fl0wer/etc/certificate.key
```

This feature is available only in the licensed version.

The evaluation version requires you to use the parameter *enable_dyncertfile* set to yes like this:

```
enable_dyncertfile = yes
```

This feature forces a dynamic certificate regeneration in memory each time the Fl0wer server is restarted, so it is strongly suggested to ***never*** put in production an evaluation release that changes its certificate data at each restart.

Note: The evaluation version, independently of the *enable_dyncert* setting, will always work as if this parameter was set to yes.

During its testing in an ISP environment, it was noted that indipendently of the used bandwidth (1 Gbit of traffic with the CPU at an average of 1.2%), the IP Cache was growing up to more than

400000 hosts per day and IP Relationships consequently reached several hundrends of thousands, slowing the interaction with the Fl0werUI. There was no packet loss, so no flows were lost, but taking info with the GUI was not so snappy, so I introduced a feature to handle this. If the Fl0wer collector has to handle tens of thousands of different IP addresses, it is strongly suggested (unless you have the CPU & Memory power to handle it) to limit tracking of IP Cache and IP Relationship information to only Internal IP addresses. This would mean that, while all incoming flows are analyzed, processed, enriched and stored, the information about hosts that are part of "INTERNET TO INTERNET" flows will not be stored in memory in the IP Cache and no IP Relationships will be stored in memory for them. The drawback is that the Scan detection module will not be able to detect "INTERNET_TO_INTERNET" scans and the hosts statistics will not be collected for this kind of traffic, but all data will be stored as usual. This feature is enabled by default and can be set by setting:

track_only_internal = yes (or no)

in the */opt/fl0wer/etc/fl0wer.conf* configuration file.

If your target with Fl0wer is to feed an Analytics software, this is a safe approach.

Further, if you want to set a limit to IP Relationships, you can do so by setting the parameter

relation_limit = 100000

in the */opt/fl0wer/etc/fl0wer.conf* configuration file. If not set, default is 100.000. Once this limit is reached, no new IP Relationships will be created and existing ones will be just updated.

Relationships

A very powerful Fl0wer feature is the fact that if properly configured (and not stressed to process too much netflow traffic), it can collect in memory the relationships between IP Address.

The feature is enabled by default using the configuration file parameter:

```
enable_relationship = yes
```

Values can be yes or no.

Flow Direction

The flow direction feature is not really a feature that can be enabled or disabled, but what it really does is to describe the topology of your network, in terms of IPv4 and IPv6 subnets.

It is basically done by several network directives in the */opt/fl0wer/etc/fl0wer.conf* file that describe your internal network (or networks in your management).

As example, a directive like:

```
network wifiguest
{
        subnet = 10.1.90.0
        netmask = 24
        description = WIFIGUEST
}
```

Will tell the Fl0wer daemon that the 10.1.90.0/24 network is an internal network, and traffic going to this network will be marked as XXXXX_TO_INTERNAL and traffic with source address in the network range will be marked as INTERNAL_TO_XXXXX (where XXXXX could be INTERNET, UNKNOWN or another INTERNAL if described).

You can as well use IPv6 network subnets if you are adopting or are moving to IPv6 to describe your subnets, like in the following example:

```
network mynet6
{
        subnet = 2001:470:615b::
        netmask = 48
        description = INTERNAL-V6
}
```

In the GUI-application you will be able to see how many flows and bytes each of your networks are using. It is obvious that the more accurate are the network definitions, the more accurate will be the numbers you will get back. A couple of charts will tell you also how much traffic every subnet is doing and what is the main direction of your traffic.

Use the examples provided in the /opt/fl0wer/etc/fl0wer.conf example configuration file.

► Hint: **The more detail you provide, the better will be the results you get.** The networks are evaluated as a first match, so the first one that matches a packet will be used. Take this into account.

NPAR

The NPAR feature is the Network Probabilistic Application Recognition. It may sound funny, but in reality it helps you to detect with a >90% precision the kind of traffic that is crossing your netflow exporters. It is Probabilistic since it bases its detection on source/destination port numbers (Netflow protocol by itself does not do Deep Packet Inspection on the traffic it sees, it only gets meta-data, except when using NBAR, but this is limited to some Cisco devices) but it works pretty well. Obviously, there are false positives (mainly return connections) but it can give you a quick idea of what kind of traffic is crossing your network. It is enabled by default and it can be enabled/disabled using the directive:

```
enable_npar = yes|no
```

Also, in the Custom Network List, the entries can be configured to change the NPAR description if the IP list is matched.

If you want to experiment with NBAR, you can enable it using the configuration file parameter:

```
enable_nbar = yes|no
```

but being incomplete, it is not actually supported. Its default value is no.

IP Cache

A very useful Fl0wer feature is the IP Cache. If enabled, Fl0wer will keep in memory all the information it learns about each IP Address it sees. This includes (but is not limited to):

- IPv4 or IPv6 Address
- FQDN (if DNS resolution is enabled)
- Flows absolute count as Source
- Packets absolute count as Source
- Bytes absolute count as Source
- Flows absolute count as Destination
- Packets absolute count as Destination
- Bytes absolute count as Destination
- Company/Organization (if information is available in the Maxmind database)
- ISP (if information is available in the Maxmind database)
- Application Traffic absolute count
- ASN numbers
- Protocol split absolute count
- TCP Flags absolute count
- GeoIP information (if information is available in the Maxmind database)
- Relations with other IP addresses

All of this information is made available to the CLI-tools and GUI-application using the provided APIs.

These options are available in the */opt/fl0wer/etc/fl0wer.conf* configuration file:

```
enable_ip_cache = yes|no
ipv4_hostcache_limit = 50000
ipv6_hostcache_limit = 50000
```

The enable_ip_cache feature is enabled by default, and the defaults for the two trees (IPv4 and Ipv6) are to limit the cache to 50000 hosts each one. Adjust them accordingly to your needs, keeping in mind that each entry uses memory.

Keep in mind that if you need very high performance, the IP Cache (with IP Relations) is probably one thing you should need to disable, since for each IP Fl0wer sees, it is going to update its statistics and data.

►Important note

Since version 1.4, the IP Cache feature, completely rewritten, keeps in memory all IP information detected data, and is purged only when a USR2 (Reset of Data in Memory) signal is sent to the Fl0wer process. If you want to limit the number of IP Addresses to keep in cache (for memory reasons), you can use the following parameters:

- *ipv4_hostcache_limit*
- *ipv6_hostcache_limit*

in the */opt/fl0wer/etc/fl0wer.conf* file.

This will prevent adding new addresses to the cache, but will still update what is already present in the cache. Just to have an idea, on a mid-size LIR ISP with a Gigabit Internet connection, we have an average of 1.300.000 different IPs per day, about 50.000.000 of different flows per day and about 1.000.000 relations per day, which sums to:

- about 2GB of Memory for IP Statistic data.
- about 1 GB of Memory for IP Relationships
- about 30Gb of Full CSV data storage per day

GeoIP

The GeoIP feature allows the user to add geographical information to the flows received using the separately available Maxmind’s GeoIP databases.

The default feature is *enable_geoip = no* since the GeoIP databases must be provided by the user.

The following features are available:

```
enable_geoip     = yes|no
dbgeo_ASNUM      = "/opt/fl0wer/geo/GeoIPASNum.dat"
dbgeo_ASNUM6     = "/opt/fl0wer/geo/GeoIPASNumv6.dat"
dbgeo_City       = "/opt/fl0wer/geo/GeoIPCity.dat"
dbgeo_ISP        = "/opt/fl0wer/geo/GeoIPISP.dat"
dbgeo_IPOrg      = "/opt/fl0wer/geo/GeoIPOrg.dat"
dbgeo_LiteCity   = "/opt/fl0wer/geo/GeoLiteCity.dat"
dbgeo_LiteCity6  = "/opt/fl0wer/geo/GeoLiteCityv6.dat"
dbgeo_Netspeed   = "/opt/fl0wer/geo/GeoIPNetSpeed.dat"
dbgeo_Country6   = "/opt/fl0wer/geo/GeoIPv6.dat"
geoip_fast       = yes|no
```

The geoip_fast works only when the enable_geoip feature is yes, and allows loading in RAM Memory the entire GeoIP databases for faster lookups. The other features simply sets the path of the corresponding database file; if a file is not available, simply put a comment (#) in front of the rule.

The currently used databases are:

- GeoIP2 and GeoIP Legacy City Database
- GeoIP2 ISP Database

If licenses for these databases are not available, the GeoLite databases can be used, at the cost of providing lot less information. Future versions *could* make use of the other MaxMind databases.

►**Important note: <u>Fl0wer uses the GeoIP Legacy database format</u>, support for GeoIP2 databases *could* be provided in future versions.**

Traffic Rules

Traffic rules are a powerful (still improving) tool that allows you to trigger actions when a certain pattern of IP traffic is matched. An example follows:

```
traffic_rule TRACKRSH
{
        exporter = any
        ipversion = any
        protocol = tcp
        src_addr = any
        src_mask = 0
        src_port = any
        dst_addr = any
        dst_mask = 0
        dst_port = 513
        maxpacketsize = any
        description = "Track rsh Tests"
        classification = suspicious
        store = yes
        action = syslog
}
```

In the above example we can see that we can apply a rule that matches for:

exporter: IPv4 or IPv6 address of a certain exporter, or *any*

ipversion: can be 4 for IPv4, 6 for IPv6 or *any*

protocol: can be tcp, udp, icmp, igmp, vrrp or *any*

src_addr: can be an IPv4 address if the ipversion is 4, an IPv6 address if the ipversion is 6 or *any*

src_mask: can be 0 to 32 in IPv4 or 0 to 128 in Ipv6. 0 means that the source address IP mask can be whatever.

dst_addr: can be an IPv4 address if the ipversion is 4, an IPv6 address if the ipversion is 6 or *any*

dst_mask: can be 0 to 32 in IPv4 or 0 to 128 in Ipv6. 0 means that the destination address IP mask can be whatever.

maxpacketsize: it is a number between 0 and 65536 and describes the maximum allowed packet size or any. The number to match this entry should be bigger or equal to the number of bytes divided the number of packets from the flow.

description: this is the description that is added in the RULE field of the flow (used in the full format binary file).

classification: can be one of normal,unexpected,anomaly,unallowed,suspicious or dangerous. NOTE: It is actually not used and reserved for future use.

store: it allows or denies storage in the binary file of the matching flow. Possible values are yes or no.

action: can be one of *syslog, mail* or *none*. If *syslog* is used, a syslog message is sent to the syslog daemon on the running system with the details of the packet. If mail is used, an e-mail alert is sent using the parameters configured in the mail section.

NOTE: Mail sending is done using separate threads to minimize delays in packet reception, but if you are sustaining a very high throughput of FPS, it is suggested to not use it.

add_to_nst: This informs the Fl0wer daemon if you want to send traffic matching this rule to the requesting GUI-application or not. Possible values are yes or no.

npar: This rule applies if the string matches the NPAR description. NOTE: This feature is still in development and could not work.

xinfo: This rule applies if the string matches the XINFO description. NOTE: This feature is still in development and could not work.

negaterule: if the negaterule value is yes, all packets NOT matching the rule will be impacted. Possible values are yes or no. NOTE: This feature is still in development and could not work.

Always remember, rules are evaluated from first to last on a first-match basis.

LUA Scripting

Fl0wer embeds a powerful LUA 5.3 scripting engine that can be activated in two cases:

1. after a flow is decoded
2. after a template is decoded

in both cases, a LUA 5.3 script can be executed allowing complete flexibility on the user side.

In the first case, a script named */opt/fl0wer/luascripts/everyflow.lua* is searched and if found (and with a valid LUA syntax) is interpreted and executed on the fly.

When executed, the following pre-defined variables are loaded with data from the flow (if the corresponding information from the flow is available, else they are blank):

flow_id:(numeric) The Flow ID Number.

flow_version:(numeric) The Netflow version this flow has been exported with.

flow_ipversion: (numeric) The IP Version of this flow (IPv4 or IPv6). Can be 4 or 6.

flow_ipexporter: (string) The IP Address of the exporter of this flow.

flow_datereceived:(numeric) The epoch (seconds from 01/01/1970 00:00:00) this flow has been received from Fl0wer.

flow_datereceived_string:(string) The flow_datereceived field formatted as string. The format is: DD/MM/YYYY HH:MM:SS

flow_sequence: (numeric) The flow sequence if available.

flow_securityRating:(numeric) Not used.

flow_trafficCategory:(string) The traffic category description as reported in the Application Identification & Classification chapter.

flow_bytes: (numeric) How many bytes where seen in this flow by the flow exporter.

flow_packets: (numeric) How many packets where seen in this flow by the flow exporter.

flow_aggregatedflows:(numeric) How many aggregated flows where seen by the flow exporter.

flow_protocol:(numeric) The Flow Protocol numeric code as per */etc/protocols*.

flow_cos:(numeric) The Flow Class Of Service.

flow_tcpflags:(numeric) The bitmap of the TCP Flags of the flow, if the protocol is 6 (TCP).

flow_srcport:(numeric) The source port of the flow.

flow_dstport:(numeric) The destination port of the flow.

flow_srcip:(string) The source IP Address of the flow. Could be an IPv4 or IPv6 address.

flow_dstip:(string) The destination IP Address of the flow. Could be an IPv4 or IPv6 address.

flow_nexthopip:(string) The IP Address of the next hop router of the flow, if available. Could be an IPv4 or IPv6 address.

flow_starttime:(numeric) The epoch (seconds from 01/01/1970 00:00:00) reported in the flow this flow started.

flow_starttime_string:(string) The flow_starttime field formatted as string. The format is: DD/MM/YYYY HH:MM:SS

flow_stoptime:(numeric) The epoch (seconds from 01/01/1970 00:00:00) reported in the flow this flow stopped.

flow_stoptime_string:(string) The flow_stoptime field formatted as string. The format is: DD/MM/YYYY HH:MM:SS

flow_npar:(string) The NPAR description of the flow.

flow_nbar:(string) The NBAR matching of the flow.

flow_rule:(string) The Traffic Rule applied to the flow.

flow_xinfo:(string) The Extra Info about the flow.

flow_direction:(string) The direction string of the flow. Could be one of:

- INTERNET_TO_INTERNAL – Traffic flow from Internet to internal network.
- INTERNAL_TO_INTERNET – Traffic flow from internal network to Internet.
- INTERNAL_TO_INTERNAL – Traffic flow from internal network to internal network.
- INTERNET_TO_INTERNET – Traffic flow from Internet to Internet.
- INTERNET_TO_UNKNOWN – Traffic flow from Internet to unknown network.

- INTERNAL_TO_UNKNOWN – Traffic flow from internal network to unknown network.
- HOST_INTERNAL – Traffic flow internal to a host (mostly coming from ipt_NETFLOW, softflowd or other internal vswitch. Hddtemp server is an example).
- UNKOWN_DIRECTION – The Fl0wer engine was not able to detect which direction has the traffic. Should never happen.

flow_src_fqdn:(string) If DNS name resolution is enabled, this strings contains the FQDN of the source IP of the flow.

flow_dst_fqdn:(string) If DNS name resolution is enabled, this strings contains the FQDN of the destination IP of the flow.

flow_cisco_username:(string) The username of the remote user as per Cisco NEL with code 40000.

flow_cisco_ingress_policy:(string) The name of the ingress policy applied by the Cisco NEL, if available.

flow_cisco_egress_policy:(string) The name of the egress policy applied by the Cisco NEL, if available.

flow_src_as:(numeric) The source Autonomous System if available.

flow_dst_as:(numeric) The destination Autonomous System if available.

flow_snmpidx_ingress:(numeric) The source SNMP index of the interface, if available.

flow_snmpidx_egress:(numeric) The destination SNMP index of the interface, if available.

flow_vlan_int:(numeric) The Source VLAN Number if available.

flow_vlan_out:(numeric) The destination VLAN Number if available.

flow_dst_mac:(string) The Source MAC Address if available.

flow_dst_mac:(string) The Destination MAC Address if available.

In the second case, a script named */opt/fl0wer/luascripts/everytemplate.lua* is searched and if found (and with a valid LUA syntax) is interpreted and executed on the fly.

When executed, the following pre-defined variables are loaded with data from the template (if the corresponding information from the template is available, else they are blank):

template_id: (numeric) Contains the template number.

template_exporterip: (string) Contains the IP address of the exporter generating this template. Can be an IPv4 or IPv6 IP Address.

template_datereceived: (numeric) The epoch (seconds from 01/01/1970 00:00:00) this template has been received from Fl0wer.

template_datereceived_string: (string) The template_datereceived field formatted as string. The format is: DD/MM/YYYY HH:MM:SS *template_fields: (numeric)* Contains how many fields are present in this template.

template_datasize: (numeric) Contains how many bytes are present in this template.

template_netflowversion: (numeric) Contains the netflow version of this template.

template_isoptionrecord: (numeric) If this template is an option record, contains 1, else 0.

template_isnbarrecord: (numeric) If this template is an NBAR record, contains 1, else 0.

The following function is available only in the template processing case:

flwDumpTemplates(stringOutputfile)

Extra Functions available in the LUA interpreter

The following functions are available in both cases in the LUA Interpreter:

- *flwSyslog(stringMessage)*: Sends to the system log the message contained in *stringMessage*
- *flwMail(stringFrom,stringTo,stringSubject,stringMessage):* Sends an email using the *stringFrom* address to the *stringTo* mailbox using the *stringSubject* as subject and *stringMessage* as the body text of the mail. Requires proper mail server configuration in the /opt/fl0wer/etc/fl0wer.conf file.

  NOTE: eMail sending is thread blocking, so use it carefully.
- *flwExec(stringCommandline)*: Executes the script contained in the string *stringCommandline*
- *flwLogError(message)*: Logs in the Fl0wer error log the string contained in the parameter *message*
- *flwGetDay(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00), it returns the numeric day.
- *flwGetMon(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00), it returns the numeric month.
- *flwGetYear(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00), it returns the numeric year.
- *flwGetHour(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00), it returns the numeric hour.
- *flwGetMin(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00), it returns the numeric minute.
- *flwGetSec(epoch)*: Given the parameter *epoch* (seconds from 01/01/1970 00:00:00) it returns the numeric second.
- *flwDateString(numericDate)*: Given the *epoch* in *numericDate*, it returns the string as a DD/MM/YYYY HH:MM:SS formatted time.

- *flwIsValidIPv4Address(ipAddress)*: Given the ipAddress string, returns 1 if it is a valid IPv4 address or 0 otherwise.
- *flwIsValidIPv6Address(ipAddress)*: Given the ipAddress string, returns 1 if it is a valid IPv6 address or 0 otherwise.
- *flwIsInternalAddress(ipAddress):* Given the ipAddress string (it should be a valid IPv4 or IPv6 address) returns 1 if it is considered an Internal IP Address or 0 otherwise.

Probably, depending on user demand, more functions will be made available.

►NOTE: If the os.exit() LUA system function is used, Fl0wer daemon will abort. If you need to exit the script before it ends, use the return statement.

For more information regarding the LUA programming language, please refer to the book “Programming in Lua, Forth Edition” by Roberto Ierusalimschy or refer to http://lua.org .

The configuration parameter *luascriptdir* informs the Fl0wer daemon where it has to look for the LUA scripts.

The default value is:

```
luascriptdir = /opt/fl0wer/luascripts
```

TOR List

Apart from matching certain port numbers (TCP 9001 and TCP 9005), Fl0wer is capable to lookup if an IP address is part of the TOR network using information available at: https://torstatus.blutmagie.de/ip_list_exit.php/Tor_ip_list_EXIT.csv

The list is read at the daemon start up from the file */opt/fl0wer/iplist/torlist.txt* and contains an IPv4 address per row. If the source or destination IP address is matched in this list, the traffic is marked as CATEGORY_NETWORK_TUNNEL. As per writing of this manual, there are no known IPv6 publicly available lists.

Custom Network List

The Custom Network List is an extremely powerful tool that allows you to classify application traffic (and change the NPAR) when an IP is matched in one of the subnets that are part of this list. A customizable configuration file located by default in the path */opt/fl0wer/iplist/mynetlist.txt* is read at start up (if enabled) with all the custom defined networks. Some are already pre-loaded.

This allows you to match well known networks like Facebook, Whatsup, Twitter, LinkedIn and so on and tag the corresponding flow with the more appropriate NPAR for immediate traffic classification.

A simple example of the entries follows:

```
enable_custom_networks = yes

network vk_1
{
        subnet = 95.213.0.0
        netmask = 18
        description = "VKontakt Russian Social Network"
        use_as_npar = yes
        as = 47541
        category = CATEGORY_SOCIAL
}

network vk_2
{
        subnet = 87.240.128.0
        netmask = 18
        description = "VKontakt Russian Social Network"
        use_as_npar = yes
        as = 47541
        category = CATEGORY_SOCIAL
}

network vk_3
{
        subnet = 2a00:bdc0::
        netmask = 36
        description = "VKontakt Russian Social Network"
        use_as_npar = yes
        as = 47541
        category = CATEGORY_SOCIAL
}
```

enable_custom_networks = *yes* or *no* tells Fl0wer if it should process the custom networks list for each flow (be careful, if your network list is very big, it could slower Flows Per Second a lot, take into account that every incoming flow is matched against this list).

The above entries are named vk_1, vk_2 and vk_3 and describe some of the public IP subnets used by the Vkontakt Russian Social Network.

The fields are as follows:

subnet: it is an IPv4 or IPv6 subnet assigned to an entity

netmask: it is the netmask that limits the subnet size

description: describes what the network is/does and it is used as NPAR description if the following field is set to yes.

use_as_npar: if set to yes, when traffic that matches the above said subnet is found, the description field is used as NPAR description

as: it is the Autonomous System number. If it is not known, use 0.

category: the matching traffic will be assigned to the indicated category.

►IMPORTANT: All the entries must be placed in the */opt/fl0wer/iplist/mynetlist.txt* configuration file, including the enable_custom_networks directive.

DNS IPv4 and IPv6 Safe Server List

Fl0wer can track DNS queries and answers and match them in a couple of files that are read at the daemon start up from the file */opt/fl0wer/iplist/dns4list.txt* and */opt/fl0wer/iplist/dns6list.txt.*

These files contains an IPv4 (for the dns4list.txt) or IPv6 (for the dns6list.txt) address per row and represent the allowed DNS servers to be queried. If the source or destination IP address is matched in this list, the field XINFO traffic is marked as a well known DNS server. If traffic is detected by NPAR as DNS but the IPs are not listed here, they are reported using the relative API function.

NTP IPv4 and IPv6 Allowed Server List

Fl0wer can track NTP queries and answers and match them in a couple of files that are read at the daemon start up from the file */opt/fl0wer/iplist/ntp4list.txt* and */opt/fl0wer/iplist/ntp6list.txt.*

These files contains an IPv4 (for the ntp4list.txt) or IPv6 (for the ntp6list.txt) address per row and represent the allowed NTP servers to be queried. If the source or destination IP address is matched in this list, the field XINFO traffic is marked as a well known NTP server. If traffic is detected by NPAR as NTP but the IPs are not listed here, they are reported using the relative API function.

BGP IPv4 and IPv6 Peers Allowed List

Fl0wer can track BGP queries and answers and match them in a couple of files that are read at the daemon start up from the file */opt/fl0wer/iplist/bgp4list.txt* and */opt/fl0wer/iplist/bgp6list.txt.*

These files contains an IPv4 (for the bgp4list.txt) or IPv6 (for the bgp6list.txt) address per row and represent the allowed BGP peers to be queried. If the source or destination IP address is matched in this list, the field XINFO traffic is marked as a well known BGP server. If traffic is detected by NPAR as BGP but the IPs are not listed here, they are reported using the relative API function.

Top 100 Flows

Fl0wer can track the top 100 flows it sees since its last data cleanup (or startup) and export it over its JSON API.

To enable this feature, you should add:

```
enable_top100 = yes
```

In the /opt/fl0wer/etc/fl0wer.conf configuration file. Use no to disable it. The default is to have it enabled.

Security Last Flows

Fl0wer can track in memory the last 100 flows related to security issues. The last security flows are split into the following categories:

- Bogons IPv4 and IPv6 related traffic flows
- IP Reputation IPv4 and IPv6 related traffic flows
- Tor related traffic flows
- P2P related traffic flows
- Unclassified traffic flows
- VPN related traffic flows
- Authentication related traffic flows
- Network Tunnel related traffic flows

They are enabled in the configuration file /opt/fl0wer/etc/fl0wer.conf using the statement:

```
enable_security_lastflows = yes
```

To disable them, use “no” as the statement value.

Mail related parameters

Version 1.1 of Fl0wer finally includes support for sending e-mails. Alert emails can be sent for traffic matching traffic rules, or internally from the LUA embedded interpreter. The following parameters can be set in the /opt/fl0wer/etc/fl0wer.conf configuration file.

mail_smtpuser = “xxx” (default none): if your SMTP mail server requires authentication, here you enter the username.

mail_smtppass = “yyy” (default none): if your SMTP mail server requires authentication, here you enter the password for the above said username.

mail_smtpserver = “xxxx” (default none): Your SMTP mail server address. Can be a name (but must be resolved) or an IP address.

mail_smtpport = number (defaul 25): The port on which your SMTP mail server accepts connections for incoming mail. For clear-text SMTP it is 25, for SSL it is normally used the 465, for TLS enabled mail servers the 587 port is commonly used.

mail_smtptype = number (0,1,2) The SMTP server type. 0 means classical clear-text SMTP. 1 means SSL enabled SMTP, while 2 means TLS enabled SMTP.

mail_rules_address = string: This is the e-mail address to which the alert mails for traffic matching traffic rules will be sent.

mail_default_address = string: This is the e-mail address to which the alert mails will be sent by default.

mail_default_sender = string: This is the e-mail address from which the alert mails will be sent by default.

Network Security Features

The 1.3 release of the Fl0wer software provides a lot of features regarding Network Security, introducing and deploying the concept of “Pervasive Security”. Altough it might seem a simple marketing strategy, the following features allows you to significantly improve you network control and security, regardless of the firewall configuration.

Risk Index

A new concept in Fl0wer is the Risk Index. Fl0wer, for every flow it sees, based on certain rules, computes a risk index regarding such flow; further, it saves in memory the rationales that lead to computing such index and stores it in the IP Cache info for a host. The Risk Index is computed based on:

- scan detections (as source or target)
- ICMP floods (as source or target)
- acting or talking with a Network Bogon
- acting or talking with an IP with Bad Reputation
- policy violations (SNMP/VPN/Tunnel/DNS/NTP/BGP not in list)
- TOR Traffic
- P2P Traffic
- IRC Traffic
- non encrypted MANAGEMENT, AUTHENTICATION or MAIL traffic types

It can be seen in the Fl0werUI in the “Threat Index” tab and in the Risk Type Chart.

In the “Threat Index” tab there is also a “Risk Rate %” which is a simple percentage of Risk versus number of flows seen on that host. You could have a 100% Risk Rate with an host doing only non encrypted traffic and a lower value for a host scanning other systems, but it can be a useful indicator too.

Flow Deduplication

In complex Netflow/IPFIX topologies, it can frequently happen that the same flow is seen (and recorded) by different Exporters. While this can be useful in several contexts, most people think that storing the flow once is more than ok (specially when using volume-oriented priced services or softwares like Splunk). Deduplication can be enabled or disabled in the configuration file using:

```
enable_deduplication = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default. It works using the next-hop field concept. Every flow (IPv4 and IPv6) in each of the supported standards can have a “next-hop” field where the Flow Exporter stores the IP address of the next-hop. If such IP is in the list of the known flow-exporters (dynamically learnt by Fl0wer), then the flow will be processed by such exporter. If it is not available or the IP is not known, the flow is stored anyway. Improvements of 30% to 50%

flows reduction has been seen in environments with flows crossing multiple exporter devices. Anyway, a good planning of flow exporters topology can bypass the need for this feature.

Network Bogons

This feature allows you to know (and react if needed) if any of your internal hosts is doing uncontrolled traffic from or to a not-assigned IANA IPv4 or IPv6 Prefix. The IANA Bogons prefix list is loaded at Fl0wer daemon startup, but can be reloaded any time as new updates appear. A good cron job can automate completely this activity. Note: since IP Bogon Prefix lists can be very large, they are loaded in background in a separate thread, and IP reputation processing will start as soon as the lists are completely loaded, while normal processing is done immediately. Depending on your CPU and IP Bogon Prefix lists size, you can see your CPU jump to 100% usage when loading the lists: this is absolutely normal since Fl0wer builds a tree in memory with all the subnets to have them readily available for processing. The fixed-path files that will be loaded are:

- */opt/fl0wer/iplist/bogonsv4.txt* - for IPv4 bogon IP Prefixes
- */opt/fl0wer/iplist/bogonsv4.txt* - for IPv6 bogon IP Prefixes

The IP Bogons checking feature is enabled using the configuration statement:

```
enable_bogons = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

IP Reputation

This feature allows you to load a list of IPv4 and IPv6 IP Prefixes (Addresses or subnets) that are considered harmful for several reasons (P2P hosts, TOR nodes, Malware repositories, Ransomware target hosts, etc). You can decide freely which one of the publicly available lists use and automate the update process using a cron job. If your firewalling infrastructure is remotely programmable, you can even automate the update policy deployment, improving the filtering capabilities even if the firewall subscription service expired. Note: since IP reputation lists can be very large, they are loaded in background in a separate thread, and IP reputation processing will start as soon as the list is completely loaded, while normal Fl0wer processing starts immediately. Depending on your CPU and reputation list size, you can see your CPU jump to 100% usage when loading the lists: this is absolutely normal since Fl0wer builds a tree in memory with all the subnets to have them readily available for processing. In /opt/fl0wer/scripts you will find an example script that downloads some IP reputation lists preparing them to be used. The fixed-path files that will be loaded are:

- */opt/fl0wer/iplist/reputationv4.txt* - for IPv4 Addresses with bad reputation
- */opt/fl0wer/iplist/reputationv6.txt* - for IPv6 Addresses with bad reputation

The IP Reputation checking feature is enabled using the configuration statement:

```
enable_reputation = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Network Scans and Floods

I have implemented a simple version of the scan and flood detection module. It is able to detect TCP SYN, XMAS, FIN and NULL scans and ICMP floods, both for IPv4 and IPv6 protocols.

Future releases will see more scans and floods detection improvements. However, with what is already available, you can have a good idea of what’s happening on the network, when, and how in an instant way.

The Scan Module is enabled using the configuration statement:

```
enable_detect_scans = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

TOR Flows

TOR is an excellent service to ensure anonymity of people in several contexts, but there are also reasons to track this kind of traffic since it is possibile that your security polices should deny this. Using a mix of publicly available TOR exit nodes list and NPAR detection, you can have a good idea about if and who is using this service. The TOR Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

P2P Flows

P2P traffic can be used for several reasons, some legal like software distribution, some illegal like piracy. NPAR allows you to detect this kind of traffic and act accordingly. The P2P Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Unclassified Flows

The NPAR detection engine is very powerful but for obvious reasons cannot detect everything. It is interesting to see what is not properly detected to act accordingly. The Unclassified Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```
in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

VPN Flows

A lot of Network Managers use NAT to provide DIA (Direct Internet Access) to internal users. While this works perfectly in most cases, it is interesting to know if there are any outbound channels like VPNs, since theoretically they could be paths to do data exfiltration. The VPN Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Authentication Flows

An often seen network design flaw is to allow authentication flows crossing the firewall and go wildly over the Internet. You can track this easily now. The Authentication Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Network Tunnel Flows

A lot of Network Managers use NAT to provide DIA (Direct Internet Access) to internal users. While this works perfectly in most cases, it is interesting to know if there are any outbound channels (like IPv4 to IPv6, proxies, etc.) since theoretically they could be paths to do data exfiltration. The Network Tunnel Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Top 100 Flows

The Top 100 Flows features allows you to know the top 100 Flows since data collection ordered by:

- Flow size (in bytes)
- Flow packets
- Flow duration (in seconds)

The Top 100 Flows feature is enabled using the configuration statement:

```
enable_top100 = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

External Management Flows

An often seen network design flaw is to allow remote management of internal systems or networks directly from the Internet (or viceversa). A VPN should be always used for this. You can track this easily now. The External Management Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Risk Flows

Apart from HTTP, DNS and QUIC protocol, any communication going to or from the Internet should be encrypted. Supposedly unencrypted flows are reported here. The Risk Flows feature is part of a feature pack called Last Flows.

The Last Flows feature is enabled using the configuration statement:

```
enable_security_lastflows = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Unknown DNS

Often internal users suffer of misconfiguration problems that are mostly due to the use of external DNS servers that don't know how to answer to internal hosts requests. You can track this easily now.

Unknown NTP

Often internal users suffer of misconfiguration problems that are mostly due to the use of external NTP servers that are not aligned to your internal network date and time. You can track this easily now.

Unknown BGP

BGP pairings with upstream nodes (internal or with ISPs) should be carefully monitored and this feature allows you to track all access attempts to a key component of your network infrastructure.

Long lived ICMP Flows

New in the 1.2 version, this feature allows to see the last 1000 long lived ICMP flows. Normally, an ICMP flow should last just a few seconds (a ping, an UDP port unreachable or information about PMTUD). Long lived channels could hide ICMP Tunnels. This feature shows the last 1000 ICMP Flows with duration greater than 3 minutes.

Network Discovery Features

The 1.2 release of the Fl0wer software introduces several features regarding information about the systems it can see on the network. Network flows are a real proof that traffic between entities happened (or, at least, it tried), so a lot of information can be used to:

- track incorrect configurations
- track misuse of resources
- track out of policy services really happening

RIP Routers

This feature allows you to know if there are any routers or systems running the RIP protocol to learn/distribute routes (in the Fl0wer visibility range).

The RIP Routers feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

OSPF Routers

This feature allows you to know if there are any routers or systems running the OSPF protocol to learn/distribute routes (in the Fl0wer visibility range).

The OSPF Routers feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

EIGRP Routers

This feature allows you to know if there are any routers or systems running the OSPF protocol to learn/distribute routes (in the Fl0wer visibility range).

The EIGRP Routers feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

BGP Routers

This feature allows you to know if there are any routers or systems running the BGP protocol to learn/distribute routes (in the Fl0wer visibility range).

The EIGRP Routers feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Netflow Learned Routers

This feature allows you to know all the “default routers” exported by Netflow/IPFIX Exporters in the Fl0wer visibility range.

The Netflow Learned Routers feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

SNMP Clients

This feature allows you to know if there are clients trying to obtain SNMP Information in the network (in the Fl0wer visibility range). In this list, you should see only authorized Network Management systems.

The SNMP Clients feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

SNMP Agents

This feature allows you to know which are the SNMP managed systems in the network (in the Fl0wer visibility range).

The SNMP Agents feature is part of a group of features that discover routing protocols Information. This feature is enabled by setting:

```
enable_discovery_router = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

CIFS Servers

This feature allows you to know if there are any systems running the CIFS/SMB protocol to share files between hosts (in the Fl0wer visibility range).

The CIFS Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_storage = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

NFS Servers

This feature allows you to know if there are any systems running the NFS protocol to share files between hosts (in the Fl0wer visibility range).

The NFS Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_storage = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

iSCSI Targets

This feature allows you to know if there are any systems running the iSCSI protocol to share LUNs between hosts (in the Fl0wer visibility range).

The iSCSI Targets feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_storage = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

FTP Servers

This feature allows you to know if there are any systems running the FTP protocol to share files between hosts (in the Fl0wer visibility range).

The FTP Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_storage = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

TFTP Servers

This feature allows you to know if there are any systems running the TFTP protocol to share files between hosts (in the Fl0wer visibility range).

The TFTP Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_storage = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

HTTP Servers

This feature allows you to know all the systems running the HTTP protocol that have been contacted in some way (in the Fl0wer visibility range).

The HTTP Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_web = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

HTTPS Servers

This feature allows you to know all the systems running the HTTPS protocol that have been contacted in some way (in the Fl0wer visibility range).

The HTTPS Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_web = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Social Network Servers

This feature allows you to know all the Social Network Servers that have been contacted in some way (in the Fl0wer visibility range).

The Social Network Servers feature is part of a group of features that discover data storage servers. This feature is enabled by setting:

```
enable_discovery_web = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

SMTP Servers

This feature allows you to know all the SMTP Servers (on port 25) that have been contacted in some way (in the Fl0wer visibility range).

The SMTP Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

SMTPS Servers

This feature allows you to know all the SMTPS (SSL&TLS on ports 587 and 465) Servers that have been contacted in some way (in the Fl0wer visibility range).

The SMTPS Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

POP3 Servers

This feature allows you to know all the POP3 (on port 110) Servers that have been contacted in some way (in the Fl0wer visibility range).

The POP3 Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

POP3S Servers

This feature allows you to know all the POP3S (on port 995) Servers that have been contacted in some way (in the Fl0wer visibility range).

The POP3S Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

IMAP Servers

This feature allows you to know all the IMAP (on port 143) Servers that have been contacted in some way (in the Fl0wer visibility range).

The IMAP Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

IMAPS Servers

This feature allows you to know all the IMAPS (on port 993) Servers that have been contacted in some way (in the Fl0wer visibility range).

The IMAPS Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Webmail Servers

This feature allows you to know all the Webmail (IP/Subnets tagged as MAIL in the Custom Networks) Servers that have been contacted in some way (in the Fl0wer visibility range).

The Webmail Servers feature is part of a group of features that discover mail servers. This feature is enabled by setting:

```
enable_discovery_mail = yes|no
```

in the /opt/fl0wer/etc/fl0wer.conf and it is enabled by default.

Security Considerations

When a service needs to be put into production, it is always necessary to consider the appropriate security measures.

The Fl0wer collector should be protected in the following ways:

- on a network side basis
- on a host side basis

The Network side

The Fl0wer server, by default, listens on 2 user-customizable ports:

- UDP/2056 to receive incoming flows from Flow Exporters
- TCP/7443 to handle requests from CLI tools and GUI Application

The system administrator needs to make the best effort to be sure that the UDP/2056 port receives data only from allowed Flow Exporters. Fl0wer has no means to know that a packet is coming from an authorized or well known Flow Exporter, and understanding the Netflow protocols, this could not be different. Using a host based firewall on the Fl0wer server, allowing only the well known exporters is a good first approach. Making sure that spoofed packets cannot reach the Fl0wer server is an even better approach.

On the other side, the so called “clientport” on TCP/7443 is a bit more secure, since it uses:

- TLSv1.2 (or SSLv2 in the worst case) with AES-256-CBC to encrypt all data communications between the Fl0wer server and the clients
- Every operation must be correctly authenticated, and passwords never travel the network in clear-text but are encrypted using SHA-512 digests travelling inside the TLSv1.2 (or SSLv2 in the worst case) encrypted channel.

Anyway, even with these security measures, Fl0wer should **never** be exposed directly to dangerous networks like the Internet. The greatest possible care has been taken to be sure that the codebase is not subject to vulnerabilities, but on the other side, I hear everyday of horror stories regarding network security, so the most conservative (and secure) approach should be always taken. If you really need to access your Fl0wer server remotely, use a VPN solution like IPSec or OpenVPN.

The Host side

Fl0wer runs on a standard Linux (or UNIX) platform; best security practices should be applied for the system management. System Hardening and Minimization are something that should be part of any decent security policy in every company that has an I/T group. System auditing should be done on a regular basis and all logs (system logs and Fl0wer ones) should be checked to be sure that critical information like the Network Infrastructure data contained in Fl0wer is accessed only by authorized people. Fl0wer runs as an unprivileged user and should never be run with root privileges.

Although no checks on the certificate are currently made by the GUI Application or the CLI tools, it is strongly suggested to never put in production an evaluation version, due to the fact that certificate for the TLSv1.2 JSON API is generate dynamically, so you cannot control certificate validity and could be subject to a MITM attack to steal your Fl0wer credentials (user & password).

The Raw file format

The binary file format used by Fl0wer is pretty simple and fast to decode.

It has mainly 2 parts:

1. a simple header
2. the records with data

The header serves just as a placeholder to know which type of encoding is used for the record data with a small signature in front. It is documented in the /opt/fl0wer/include/public_interface.h file and the C structure describing it is:

```
typedef struct netflowFileHeader
{
        type_ubyte1      marker[2];
        type_ubyte1      recordFormat;
        type_ubyte1      version;
} netflowFileHeader;
```

The marker field must contain the 2 letters “GP”, while the recordFormat field can be one of:

```
#define STORAGE_DEMO                    3
#define STORAGE_COMPACT                 4
#define STORAGE_ANALYTIC                5
#define STORAGE_FULL                    6
#define STORAGE_CSVFULL                 7
```

DEMO Data Structure

When the STORAGE_DEMO model is used, the records with data section is composed by 1 or more of the following C structure:

```
typedef struct recordDemoFormat
{
        type_ubyte1      filler1[8192];
        type_ulong8      flow_id;
        type_ubyte1      Netflow_Type;
        type_ubyte1      IP_Version;
        type_ubyte1      IP_Src_FlowExporter[41];
        type_int4        dateReceived;
        type_uint4       ID_001_byteDeltaCount;
        type_uint4       ID_002_packetDeltaCount;
        type_ubyte1      ID_004_protocolIdentifier;
        type_ubyte1      ID_005_ipClassOfService;
        type_ubyte1      ID_006_tcpControlBits;
        type_ushort2     ID_007_sourceTransportPort;
        type_ubyte1      ID_008_sourceIPAddress[41];
        type_ushort2     ID_011_destinationTransportPort;
        type_ubyte1      ID_012_destinationIPAddress[41];
        type_uint4       ID_016_bgpSourceAsNumber;
        type_uint4       ID_017_bgpDestinationAsNumber;
        type_int4        ID_021_flowEndSysUpTime;
        type_int4        ID_022_flowStartSysUpTime;
        type_ubyte1      nexthopIPAddress[41];
        type_ubyte1      filler2[8192];
} recordDemoFormat;
```

This format is used only in the public demo version.

COMPACT Data Structure

When the STORAGE_COMPACT model is used, the records with data section is composed by 1 or more of the following C structure:

```
typedef struct recordCompactFormat
{
        type_ulong8      flow_id;
        type_ubyte1      Netflow_Type;
        type_ubyte1      IP_Version;
        type_ubyte1      IP_Src_FlowExporter[41];
        type_int4        dateReceived;
        type_uint4       ID_001_byteDeltaCount;
        type_uint4       ID_002_packetDeltaCount;
        type_ubyte1      ID_004_protocolIdentifier;
        type_ubyte1      ID_005_ipClassOfService;
        type_ubyte1      ID_006_tcpControlBits;
        type_ushort2     ID_007_sourceTransportPort;
        type_ubyte1      ID_008_sourceIPAddress[41];
        type_ushort2     ID_011_destinationTransportPort;
        type_ubyte1      ID_012_destinationIPAddress[41];
        type_uint4       ID_016_bgpSourceAsNumber;
        type_uint4       ID_017_bgpDestinationAsNumber;
        type_int4        ID_021_flowEndSysUpTime;
        type_int4        ID_022_flowStartSysUpTime;
        type_ubyte1      nexthopIPAddress[41];
} recordCompactFormat;
```

ANALYTIC Data Structure

When the STORAGE_ANALYTIC model is used, the records with data section is composed by 1 or more of the following C structure:

```
typedef struct recordAnalyticFormat
{
        type_ulong8      flow_id;
        type_ubyte1      Netflow_Type;
        type_ubyte1      IP_Version;
        type_ubyte1      IP_Src_FlowExporter[41];
        type_int4        dateReceived;
        type_uint4       flowSequence;
        type_ubyte1      trafficCategory;
        type_ubyte1      IPFlowDirection;
        type_uint4       ID_001_byteDeltaCount;
        type_uint4       ID_002_packetDeltaCount;
        type_ubyte1      ID_004_protocolIdentifier;
        type_ubyte1      ID_005_ipClassOfService;
        type_ubyte1      ID_006_tcpControlBits;
        type_ushort2     ID_007_sourceTransportPort;
        type_ubyte1      ID_008_sourceIPAddress[41];
        type_ushort2     ID_011_destinationTransportPort;
        type_ubyte1      ID_012_destinationIPAddress[41];
        type_uint4       ID_016_bgpSourceAsNumber;
        type_uint4       ID_017_bgpDestinationAsNumber;
        type_int4        ID_021_flowEndSysUpTime;
        type_int4        ID_022_flowStartSysUpTime;
        type_ubyte1      nexthopIPAddress[41];
        type_ubyte1      NPAR[256];
} recordAnalyticFormat;
```

FULL Data Structure

When the STORAGE_FULL model is used, the records with data section is composed by 1 or more of the following C structure:

```
typedef struct recordFullFlowFormat
{
        type_ulong8      flow_id;
        type_ubyte1      Netflow_Type;
        type_ubyte1      IP_Version;
        type_ubyte1      IP_Src_FlowExporter[41];
        type_int4        dateReceived;
        type_uint4       flowSequence;
        type_ubyte1      trafficCategory;
        type_ubyte1      securityRating;
        type_ushort2     riskLevel;
        type_ushort2     riskType;
        type_ubyte1      classification;
        type_ulong8      FlowDuration;
        type_ubyte1      IPFlowDirection;

        type_uint4       ID_001_byteDeltaCount;
        type_uint4       ID_002_packetDeltaCount;

        type_uint4       ID_003_aggregatedFlows;

        type_ubyte1      ID_004_protocolIdentifier;
        type_ubyte1      ID_005_ipClassOfService;
        type_ubyte1      ID_006_tcpControlBits;
        type_ushort2     ID_007_sourceTransportPort;
        type_ubyte1      ID_008_sourceIPAddress[41];
        type_ubyte1      ID_009_sourceIPv4PrefixLength;
        type_ushort2     ID_010_ingressInterface;
        type_ushort2     ID_011_destinationTransportPort;
        type_ubyte1      ID_012_destinationIPAddress[41];
        type_ubyte1      ID_013_destinationIPv4PrefixLength;
        type_ushort2     ID_014_egressInterface;

        type_ubyte1      nexthopIPAddress[41];
```

```
type_uint4       ID_016_bgpSourceAsNumber;
type_uint4       ID_017_bgpDestinationAsNumber;

type_ubyte1      ID_018_bgpNextHopIPAddress[41];
type_uint4       ID_019_postMCastPacketDeltaCount;
type_uint4       ID_020_postMCastbyteDeltaCount;

type_int4        ID_021_flowEndSysUpTime;
type_int4        ID_022_flowStartSysUpTime;

type_uint4       ID_023_postbyteDeltaCount;
type_uint4       ID_024_postPacketDeltaCount;
type_ushort2     ID_025_minimumIpTotalLength;
type_ushort2     ID_026_maximumIpTotalLength;
type_ubyte1      ID_029_sourceIPv6PrefixLength;
type_ubyte1      ID_030_destinationIPv6PrefixLength;
type_uint4       ID_031_flowLabelIPv6;
type_ushort2     ID_032_icmpTypeCodeIPv4;
type_ubyte1      ID_033_igmpType;
type_uint4       ID_034_samplingInterval;
type_ubyte1      ID_035_samplingAlgorithm;
type_ushort2     ID_036_flowActiveTimeout;
type_ushort2     ID_037_flowIdleTimeout;
type_ubyte1      ID_038_engineType;
type_ubyte1      ID_039_engineID;
type_uint4       ID_040_exportedbyteTotalCount;
type_uint4       ID_041_exportedMessageTotalCount;
type_uint4       ID_042_exportedFlowRecordTotalCount;
type_uint4       ID_044_sourceIPv4Prefix;
type_uint4       ID_045_destinationIPv4Prefix;
type_ubyte1      ID_046_mplsTopLabelType;
type_ubyte1      ID_047_mplsTopLabelIPv4Address[18];
type_ubyte1      ID_048_flowSamplerID;
type_ubyte1      ID_049_flowSamplerMode;
type_uint4       ID_050_flowSamplerRandomInterval;
type_ubyte1      ID_052_minimumTTL;
type_ubyte1      ID_053_maximumTTL;
type_ushort2     ID_054_fragmentIdentification;
type_ubyte1      ID_055_postIpClassOfService;
type_ubyte1      ID_056_sourceMacAddress[20];
type_ubyte1      ID_057_postDestinationMacAddress[20];
type_ushort2     ID_058_vlanId;
type_ushort2     ID_059_postVlanId;

type_ubyte1      ID_080_destinationMacAddress[20];
type_ubyte1      ID_081_postSourceMacAddress[20];
type_ubyte1      ID_082_interfaceName[32];
type_ubyte1      ID_083_interfaceDescription[64];

type_uint4       ID_085_byteTotalCount;
type_uint4       ID_086_packetTotalCount;
type_ushort2     ID_088_fragmentOffset;
type_uint4       ID_089_forwardingStatus;
type_ulong8      ID_090_mplsVpnRouteDistinguisher;
type_ubyte1      ID_091_mplsTopLabelPrefixLength;
type_uint4       ID_092_srcTrafficIndex;
type_uint4       ID_093_dstTrafficIndex;
type_ubyte1      ID_098_postIpDiffServCodePoint;
type_uint4       ID_099_multicastReplicationFactor;

type_ubyte1      ID_101_classificationEngineId;
type_uint4       ID_102_layer2PacketSectionOffset;
type_uint4       ID_103_layer2PacketSectionSize;
type_uint4       ID_104_layer2PacketSectionData;
type_ushort2     ID_128_bgpNextAdjacentAsNumber;
type_ushort2     ID_129_bgpPrevAdjacentAsNumber;

type_ubyte1      ID_136_flowEndReason;
type_ulong8      ID_137_commonPropsID;
type_ulong8      ID_138_observationPointID;
type_ushort2     ID_139_icmpTypeCodeIPv6;
type_ubyte1      ID_140_mplsTopLabelIPv6Address[41];
type_uint4       ID_141_lineCardID;
type_uint4       ID_142_portID;
type_uint4       ID_143_meteringProcessID;
type_uint4       ID_144_exportingProcessID;
type_ushort2     ID_145_templateID;
```

```
type_ubyte1      ID_146_wlanChannelID;
type_ubyte1      ID_147_wlanSSID[40];
type_ulong8      ID_148_cisco_asa_NF_F_CONN_ID;
type_uint4       ID_149_observationDomainID;
type_ulong8      ID_163_ObservedFlowtotalCount;
type_ubyte1      ID_176_cisco_asa_NF_F_ICMP_TYPE;
type_ubyte1      ID_177_cisco_asa_NF_F_ICMP_CODE;
type_ubyte1      ID_178_cisco_asa_NF_F_ICMP_TYPE_IPV6;
type_ubyte1      ID_179_cisco_asa_NF_F_ICMP_CODE_IPV6;
type_uint4       ID_184_tcpSequence;
type_ubyte1      ID_192_ipTTL;
type_ubyte1      ID_195_ipDiffServCodePoint;
type_ubyte1      ID_196_ipPrecedence;
type_ubyte1      ID_197_fragmentFlags;
type_ubyte1      ID_200_mplsTopLabelTTL;
type_uint4       ID_208_ipv4Options;
type_uint4       ID_209_TCP_Option_Map;
type_ulong8      ID_224_ipTotalLength;
type_ubyte1      ID_225_cisco_asa_NF_F_XLATE_SRC_ADDR_IPV4[18];
type_ubyte1      ID_226_cisco_asa_NF_F_XLATE_DST_ADDR_IPV4[18];
type_ushort2     ID_227_cisco_asa_NF_F_XLATE_SRC_PORT;
type_ushort2     ID_228_cisco_asa_NF_F_XLATE_DST_PORT;
type_byte1       ID_230_natEvent;
type_ubyte1      ID_281_cisco_asa_NF_F_XLATE_SRC_ADDR_IPV6[41];
type_ubyte1      ID_282_cisco_asa_NF_F_XLATE_DST_ADDR_IPV6[41];
type_ubyte1      ID_233_cisco_asa_NF_F_FW_EVENT;
type_uint4       ID_234_ingressVRFID;
type_uint4       ID_235_egressVRFID;
type_ushort2     ID_33002_cisco_asa_NF_F_FW_EXT_EVENT;
type_ulong8      ID_323_cisco_asa_NF_F_EVENT_TIME_MSEC;
type_ulong8      ID_152_cisco_asa_NF_F_FLOW_CREATE_TIME_MSEC;
type_uint4       ID_231_cisco_asa_NF_F_FWD_FLOW_DELTA_BYTES;
type_uint4       ID_232_cisco_asa_NF_F_REV_FLOW_DELTA_BYTES;
type_ubyte1      ID_33000_cisco_asa_NF_F_INGRESS_ACL_ID[33];
type_ubyte1      ID_33001_cisco_asa_NF_F_EGRESS_ACL_ID[33];
type_ubyte1      ID_40000_cisco_asa_NF_F_USERNAME[32];
type_ubyte1      ID_40001_cisco_asa_NF_F_XLATE_SRC_ADDR_IPV4[18];
type_ubyte1      ID_40002_cisco_asa_NF_F_XLATE_DST_ADDR_IPV4[18];
type_ushort2     ID_40003_cisco_asa_NF_F_XLATE_SRC_PORT;
type_ushort2     ID_40004_cisco_asa_NF_F_XLATE_DST_PORT;
type_ubyte1      ID_40005_cisco_asa_NF_F_FW_EVENT;

type_ubyte1      src_organization[100];
type_ubyte1      src_beginip[18];
type_ubyte1      src_endip[18];
type_ushort2     src_assigned_netmask;
type_ubyte1      src_isp_as[60];
type_ubyte1      src_countryCode[4];
type_ubyte1      src_region[20];
type_ubyte1      src_region_name[60];
type_ubyte1      src_city[60];
type_ubyte1      src_postalcode[16];
type_ubyte1      src_longitude[48];
type_ubyte1      src_latitude[48];
type_int4        src_areacode;
type_ubyte1      src_timezone[32];

type_ubyte1      dst_organization[100];
type_ubyte1      dst_beginip[18];
type_ubyte1      dst_endip[18];
type_ushort2     dst_assigned_netmask;
type_ubyte1      dst_isp_as[60];
type_ubyte1      dst_countryCode[4];
type_ubyte1      dst_region[20];
type_ubyte1      dst_region_name[60];
type_ubyte1      dst_city[60];
type_ubyte1      dst_postalcode[16];
type_ubyte1      dst_longitude[48];
type_ubyte1      dst_latitude[48];
type_int4        dst_areacode;
type_ubyte1      dst_timezone[32];

type_ubyte1      srcHostname[96];
type_ubyte1      dstHostname[96];
type_ubyte1      RULE[SIZE_RULE];
type_ubyte1      NBAR[SIZE_NBAR];
type_ubyte1      NPAR[SIZE_NPAR];
type_ubyte1      XINFO[SIZE_XINFO];
```

```
        type_ubyte1      observationDomain[32];
        type_ushort2     templateID;
} recordFullFlowFormat;
```

The field names are self-explaining, and the prefix numbers where available, refer to the corresponding Netflow V1/V5/V9/IPFIX field in official Cisco(TM) or IETF documentation.

►IMPORTANT: All numbers with a byte size greater than 1 byte are encoded using **Network Byte Order** for data portability, so, as shown in the provided examples, the correct way of decoding is using the ntohl, ntohs or ntoh64 functions. Floating numbers are stored as strings and should be decoded using the C language *atof()* family functions.

JSON API

The Fl0wer daemon JSON API is pretty simple and uses an encrypted and authenticated protocol to talk with the server.

The Client establishes a TLS v1.2 TCP session the Fl0wer daemon on TCP Port 7443 (can be changed using the *clientport* parameter in the /opt/fl0wer/etc/fl0wer.conf configuration file) and authenticates himself sending a packet containing:

- the string “User:”
- a fixed length (32 bytes) string containing the username
- the string “Auth:”
- an SHA-512 digest containing the password.

If this protocol is not followed, the Fl0wer daemon replies with a KO string and closes the socket.

If this phase is completed successfully, the client sends its requests and receives a JSON reply containing the data that is requested.

Every activity is logged by the Fl0wer daemon.

Possible requests are documented in the */opt/fl0wer/src/cli/fl0wernet.py* or */opt/fl0wer/src/client/fl0wernet.py* Python 3 scripts and resemble a bit the classical HTTP requests, but all replies are made using JSON encoding and don’t use HTTP encoding.

All requests and replies are conceived to work using the RESTful model, so are self-containing and the socket is closed after each request. Please refer to the above files for implementation details.

►**If not otherwise specified, all numeric data has to be considered absolute and refers to the start of data collection.**

If a certain type of data has never been seen and the software should provide a date (start, stop, first packet, last seen, etc.), it would be reported as the start of the UNIX Epoch, so it would be reported as ‘01/01/1970 01:00:00’.

If there is no data to report, the secure socket will simply be closed by the Fl0wer server.

All fields have self-explaining names, examples are provided to show and understand.

As per writing of this manual, the available functions are detailed in the following paragraphs.

All dates are reported as DD/MM/YYYY HH:MM:SS and this behaviour cannot be changed.

General Functions

def GetFlowerJSONData(user, password, server, port, request): This is the basic function used to interact with the Fl0wer server. The other functions are all wrappers returning the retrieved JSON data.

This function implements the authentication procedure and data retrieval as described in the previous paragraph. Data is always returned as a JSON object.

def GetFlowerInfo(user, password, server, port): This function returns general information about the given Fl0wer server.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Fl0werInfo': {'Cores': '2',
                 'DataFrom': '20/03/2017 00:01:02',
                 'Hardware': 'x86_64',
                 'OpSys': 'GNU/Linux',
                 'Release': 'Fl0wer Licensed version (R) Gilberto Persico '
                            '2017',
                 'Threading': 'MULTI-THREADING ENABLED',
                 'activeThreads': '2',
                 'hostIPv4CacheCurrent': '1957',
                 'hostIPv4CacheLimit': '50000',
                 'hostIPv4Seen': '3936',
                 'hostIPv6CacheCurrent': '965',
                 'hostIPv6CacheLimit': '50000',
                 'hostIPv6Seen': '965',
                 'requestedThreads': '2'}}]
```

def GetTrafficRules(user, password, server, port): This function returns all available information about the configured Traffic Rules.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Rule0': {'action': 'NONE',
            'addToNST': 'yes',
            'bytes': '0',
            'classification': '[NORMAL] ',
            'description': 'Mikrotik neighbour discovery protocol',
            'dstAddr': 'any',
            'dstMask': 'any',
            'dstPort': '5678',
            'exporter': 'any',
            'hits': '0',
            'ipProtocol': 'any',
            'ipVersion': 'any',
            'maxpacketsize': 'any',
            'name': 'mikrotik_discovery',
            'negateRule': 'false',
            'nparMATCH': '',
            'packets': '0',
            'srcAddr': 'any',
            'srcMask': 'any',
            'srcPort': '5678',
            'storeMatching': 'false',
            'xinfoMATCH': ''}},
 < … OMISSIS … >
 {'Rule12': {'action': 'SYSLOG',
             'addToNST': 'yes',
             'bytes': '0',
             'classification': '[SUSPICIOUS] ',
             'description': 'Track rsh Tests',
             'dstAddr': 'any',
             'dstMask': 'any',
             'dstPort': '513',
             'exporter': 'any',
             'hits': '0',
             'ipProtocol': 'TCP',
             'ipVersion': 'any',
             'maxpacketsize': 'any',
             'name': 'TRACKRSH',
             'negateRule': 'false',
             'nparMATCH': '',
             'packets': '0',
             'srcAddr': 'any',
             'srcMask': 'any',
             'srcPort': 'any',
             'storeMatching': 'true',
             'xinfoMATCH': ''}}]
```

Network Lists functions

def GetKnownNetworks(user, password, server, port): This function returns all available information about the Internal Networks (those defined in the *fl0wer.conf* configuration file).

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Network0': {'bytes': '0',
               'description': 'Tunnel HE',
               'hits': '0',
               'ipVersion': '6',
               'name': 'hurricane',
               'netmask': '64',
               'packets': '0',
               'subnet': '2001:470:1f06:1b7::2'}},
 < … OMISSIS … >
 {'Network16': {'bytes': '13637106',
                'description': 'WIFIGUEST',
                'hits': '2326',
                'ipVersion': '4',
                'name': 'wifiguest',
                'netmask': '24',
                'packets': '236997',
                'subnet': '10.1.90.0'}}]
```

def GetCustomNetworks(user, password, server, port): This function returns all available information about the Custom Networks.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'CustomNetwork0': {'as': '0',
                     'bytes': '0',
                     'description': 'Facebook',
                     'hits': '0',
                     'ipVersion': '4',
                     'name': 'fb1',
                     'netmask': '21',
                     'packets': '0',
                     'subnet': '31.13.24.0',
                     'trafficCategory': 'SOCIAL',
                     'use_as_npar': 'TRUE'}},
 < … OMISSIS … >
 {'CustomNetwork758': {'as': '16509',
                       'bytes': '0',
                       'description': 'Soundcloud Social Network',
                       'hits': '0',
                       'ipVersion': '4',
                       'name': 'soundcloud_1',
                       'netmask': '16',
                       'packets': '0',
                       'subnet': '54.192.0.0',
                       'trafficCategory': 'SOCIAL',
                       'use_as_npar': 'TRUE'}}]
```

def GetNPARTableTCP(user, password, server, port): This function returns all the TCP NPAR data definitions with all the hits (absolute count) since the start up of the Fl0wer daemon.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'tcpNPAR0': {'bytes': '0',
               'description': 'TCP 20 FTP-Data Protocol',
               'hits': '0',
               'packets': '0',
               'port': '20',
               'protocol': 'TCP',
               'trafficCategory': 'DATA STORAGE'}},
 < … OMISSIS … >
 {'tcpNPAR807': {'bytes': '0',
                 'description': 'TCP 60030 HBASE Region Server Info Port',
                 'hits': '0',
                 'packets': '0',
                 'port': '60030',
                 'protocol': 'TCP',
                 'trafficCategory': 'DATABASE'}},
 {'tcpNPAR_EOF': {}}]
```

def GetNPARTableUDP(user, password, server, port): This function returns all the UDP NPAR data definitions with all the hits (absolute count) since the start up of the Fl0wer daemon.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'udpNPAR0': {'bytes': '0',
               'description': 'UDP 42 AUTH Microsoft WINS Replica Protocol',
               'hits': '0',
               'packets': '0',
               'port': '42',
               'protocol': 'UDP',
               'trafficCategory': 'AUTHENTICATION'}},
 < … OMISSIS … >
 {'udpNPAR287': {'bytes': '4337',
                 'description': 'UDP 53413 Possible NETIS Router Backdoor',
                 'hits': '33',
                 'packets': '52',
                 'port': '53413',
                 'protocol': 'UDP',
                 'trafficCategory': 'P2P'}},
 {'udpNPAR_EOF': {}}]
```

def GetServerConf(user, password, server, port): Gets the features enabled on the Fl0wer server.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ServerConfig0': {'BGP4ListEnabled': 'True',
                    'BGP6ListEnabled': 'False',
                    'Build': '2102',
                    'CustomNetworksEnabled': 'True',
                    'DNS4ListEnabled': 'True',
                    'DNS6ListEnabled': 'True',
                    'GeoIPEnabled': 'True',
                    'IPCacheEnabled': 'True',
                    'IPCacheValid': 'True',
                    'LUAEnabled': 'True',
                    'License': 'This copy of fl0wer is licensed to '
                               'name surname address town city '
                               'country. Its use is '
                               'allowed on 1 (one) Linux Kernel image '
                               'running on Physical or Virtual Hardware.',
                    'NPAREnabled': 'True',
                    'NTP4ListEnabled': 'True',
                    'NTP6ListEnabled': 'True',
                    'RelationshipEnabled': 'True',
                    'TORListEnabled': 'True',
                    'TrafficRulesEnabled': 'False',
                    'Version': '1.0'}}]
```

def ResetData(user, password, server, port): Cleans up all data in RAM memory, including counters.

The following things are done:

- IPv4 Cache: Removal of all hosts information
- IPv6 Cache: Removal of all hosts information
- Removal of all Relationships
- Cleanup of all NPAR Statistics
- Flow Exporters: Removal of all the information
- Network Services: Cleanup of all statistics
- Custom Network List: Cleanup of all statistics
- Internal Networks: Cleanup of all statistics
- Netflow Templates: Cleanup of all statistics

Flow Exporter functions

The flow exporter APIs deal with the Netflow exporters sending Netflow data to Fl0wer.

def GetExportersV4(user, password, server, port): This functions returns all available information about the IPv4 flow exporters.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'exporter0': {'ICMPTrafficBytes': '0',
                'ICMPTrafficPackets': '0',
                'IGMPTrafficBytes': '0',
                'IGMPTrafficPackets': '0',
                'IPv4TrafficBytes': '7076685',
                'IPv4TrafficPackets': '31226',
                'IPv6TrafficBytes': '1049068',
                'IPv6TrafficPackets': '5868',
                'OtherTrafficBytes': '0',
                'OtherTrafficPackets': '0',
                'TCPTrafficBytes': '0',
                'TCPTrafficFlows': '0',
                'TCPTrafficPackets': '0',
                'UDPTrafficBytes': '8125753',
                'UDPTrafficFlows': '11590',
                'UDPTrafficPackets': '37094',
                'avgFlowSec': '0',
                'avgPktSec': '0',
                'counter_bytes_FLOW_HOST_INTERNAL': '0',
                'counter_bytes_FLOW_INTERNAL_TO_INTERNAL': '7076685',
                'counter_bytes_FLOW_INTERNAL_TO_INTERNET': '1049068',
                'counter_bytes_FLOW_INTERNAL_TO_UNKNOWN': '0',
                'counter_bytes_FLOW_INTERNET_TO_INTERNAL': '0',
                'counter_bytes_FLOW_INTERNET_TO_INTERNET': '0',
                'counter_bytes_FLOW_INTERNET_TO_UNKNOWN': '0',
                'counter_bytes_FLOW_UNKNOWN': '0',
                'counter_packets_FLOW_HOST_INTERNAL': '0',
                'counter_packets_FLOW_INTERNAL_TO_INTERNAL': '0',
                'counter_packets_FLOW_INTERNAL_TO_INTERNET': '5868',
                'counter_packets_FLOW_INTERNAL_TO_UNKNOWN': '0',
                'counter_packets_FLOW_INTERNET_TO_INTERNAL': '0',
                'counter_packets_FLOW_INTERNET_TO_INTERNET': '0',
                'counter_packets_FLOW_INTERNET_TO_UNKNOWN': '0',
                'counter_packets_FLOW_UNKNOWN': '0',
                'exporterAddress': '10.1.30.20',
                'firstPacketTime': '20/03/2017 00:01:04',
                'lastPacketTime': '20/03/2017 11:58:55',
                'nf10flows': '0',
                'nf10packets': '0',
                'nf10templates': '0',
                'nf1flows': '0',
                'nf1packets': '0',
                'nf5flows': '0',
                'nf5packets': '0',
                'nf9flows': '11590',
                'nf9packets': '5069',
                'nf9templates': '482',
                'tcp_ACK_count': '0',
                'tcp_CWR_count': '0',
                'tcp_ECN_count': '0',
                'tcp_FIN_count': '0',
                'tcp_PSH_count': '0',
                'tcp_RST_count': '0',
                'tcp_SYN_count': '0',
                'tcp_URG_count': '0',
                'totBytesSeen': '8125753',
                'totPacketsSeen': '37094',
                'trafficAuthentication': '0',
                'trafficDataStorage': '0',
                'trafficDatabase': '0',
                'trafficGaming': '0',
                'trafficMAIL': '0',
                'trafficManagement': '5887348',
                'trafficNetworkOperation': '2238405',
                'trafficNetworkTunnel': '0',
                'trafficP2P': '0',
                'trafficPrinting': '0',
                'trafficSocial': '0',
                'trafficSystemsOperation': '0',
                'trafficUnclassified': '0',
                'trafficVPN': '0',
                'trafficVoiceVideo': '0',
                'trafficWEB': '0'}},
< … OMISSIS … >
 {'exporter14': {'ICMPTrafficBytes': '0',
                 'ICMPTrafficPackets': '0',
                 'IGMPTrafficBytes': '920',
                 'IGMPTrafficPackets': '20',
                 'IPv4TrafficBytes': '58568',
                 'IPv4TrafficPackets': '729',
                 'IPv6TrafficBytes': '0',
                 'IPv6TrafficPackets': '0',
                 'OtherTrafficBytes': '44496',
                 'OtherTrafficPackets': '615',
                 'TCPTrafficBytes': '0',
                 'TCPTrafficFlows': '0',
                 'TCPTrafficPackets': '0',
                 'UDPTrafficBytes': '13152',
                 'UDPTrafficFlows': '89',
                 'UDPTrafficPackets': '94',
                 'avgFlowSec': '0',
                 'avgPktSec': '0',
                 'counter_bytes_FLOW_HOST_INTERNAL': '0',
                 'counter_bytes_FLOW_INTERNAL_TO_INTERNAL': '13152',
                 'counter_bytes_FLOW_INTERNAL_TO_INTERNET': '45416',
                 'counter_bytes_FLOW_INTERNAL_TO_UNKNOWN': '0',
                 'counter_bytes_FLOW_INTERNET_TO_INTERNAL': '0',
                 'counter_bytes_FLOW_INTERNET_TO_INTERNET': '0',
                 'counter_bytes_FLOW_INTERNET_TO_UNKNOWN': '0',
                 'counter_bytes_FLOW_UNKNOWN': '0',
                 'counter_packets_FLOW_HOST_INTERNAL': '0',
                 'counter_packets_FLOW_INTERNAL_TO_INTERNAL': '0',
                 'counter_packets_FLOW_INTERNAL_TO_INTERNET': '635',
                 'counter_packets_FLOW_INTERNAL_TO_UNKNOWN': '0',
                 'counter_packets_FLOW_INTERNET_TO_INTERNAL': '0',
                 'counter_packets_FLOW_INTERNET_TO_INTERNET': '0',
                 'counter_packets_FLOW_INTERNET_TO_UNKNOWN': '0',
                 'counter_packets_FLOW_UNKNOWN': '0',
                 'exporterAddress': '10.100.5.123',
                 'firstPacketTime': '20/03/2017 00:21:47',
                 'lastPacketTime': '20/03/2017 11:59:51',
                 'nf10flows': '0',
                 'nf10packets': '0',
                 'nf10templates': '0',
                 'nf1flows': '0',
                 'nf1packets': '0',
                 'nf5flows': '0',
                 'nf5packets': '0',
                 'nf9flows': '561',
                 'nf9packets': '991',
                 'nf9templates': '430',
                 'tcp_ACK_count': '0',
                 'tcp_CWR_count': '0',
                 'tcp_ECN_count': '0',
                 'tcp_FIN_count': '0',
                 'tcp_PSH_count': '0',
                 'tcp_RST_count': '0',
                 'tcp_SYN_count': '0',
                 'tcp_URG_count': '0',
                 'totBytesSeen': '58568',
                 'totPacketsSeen': '729',
                 'trafficAuthentication': '0',
                 'trafficDataStorage': '0',
                 'trafficDatabase': '0',
                 'trafficGaming': '0',
                 'trafficMAIL': '0',
                 'trafficManagement': '11936',
                 'trafficNetworkOperation': '46632',
                 'trafficNetworkTunnel': '0',
                 'trafficP2P': '0',
                 'trafficPrinting': '0',
                 'trafficSocial': '0',
                 'trafficSystemsOperation': '0',
                 'trafficUnclassified': '0',
                 'trafficVPN': '0',
                 'trafficVoiceVideo': '0',
                 'trafficWEB': '0'}},
 {'exporter15': {'exporterAddress': 'EOL'}}]
```

def GetExportersV6(user, password, server, port): This functions returns all available information about the IPv6 flow exporters.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'exporter0': {'ICMPTrafficBytes': '0',
                'ICMPTrafficPackets': '0',
                'IGMPTrafficBytes': '552',
                'IGMPTrafficPackets': '12',
                'IPv4TrafficBytes': '6345863',
                'IPv4TrafficPackets': '50768',
                'IPv6TrafficBytes': '306784',
                'IPv6TrafficPackets': '3678',
                'OtherTrafficBytes': '223876',
                'OtherTrafficPackets': '3241',
                'TCPTrafficBytes': '46',
                'TCPTrafficFlows': '1',
                'TCPTrafficPackets': '1',
                'UDPTrafficBytes': '6428173',
                'UDPTrafficFlows': '1258',
                'UDPTrafficPackets': '6428173',
                'avgFlowSec': '0',
                'avgPktSec': '0',
                'counter_bytes_FLOW_HOST_INTERNAL': '0',
                'counter_bytes_FLOW_INTERNAL_TO_INTERNAL': '4591165',
                'counter_bytes_FLOW_INTERNAL_TO_INTERNET': '2059762',
                'counter_bytes_FLOW_INTERNAL_TO_UNKNOWN': '0',
                'counter_bytes_FLOW_INTERNET_TO_INTERNAL': '264',
                'counter_bytes_FLOW_INTERNET_TO_INTERNET': '1456',
                'counter_bytes_FLOW_INTERNET_TO_UNKNOWN': '0',
                'counter_bytes_FLOW_UNKNOWN': '0',
                'counter_packets_FLOW_HOST_INTERNAL': '0',
                'counter_packets_FLOW_INTERNAL_TO_INTERNAL': '0',
                'counter_packets_FLOW_INTERNAL_TO_INTERNET': '23367',
                'counter_packets_FLOW_INTERNAL_TO_UNKNOWN': '0',
                'counter_packets_FLOW_INTERNET_TO_INTERNAL': '3',
                'counter_packets_FLOW_INTERNET_TO_INTERNET': '28',
                'counter_packets_FLOW_INTERNET_TO_UNKNOWN': '0',
                'counter_packets_FLOW_UNKNOWN': '0',
                'exporterAddress': '2001:470:8b5b:0:10:1:30:143',
                'firstPacketTime': '20/03/2017 00:02:01',
                'lastPacketTime': '20/03/2017 12:00:01',
                'nf10flows': '0',
                'nf10packets': '0',
                'nf10templates': '0',
                'nf1flows': '0',
                'nf1packets': '0',
                'nf5flows': '0',
                'nf5packets': '0',
                'nf9flows': '1505',
                'nf9packets': '341',
                'nf9templates': '37',
                'tcp_ACK_count': '1',
                'tcp_CWR_count': '0',
                'tcp_ECN_count': '0',
                'tcp_FIN_count': '0',
                'tcp_PSH_count': '0',
                'tcp_RST_count': '0',
                'tcp_SYN_count': '0',
                'tcp_URG_count': '0',
                'totBytesSeen': '6652647',
                'totPacketsSeen': '54446',
                'trafficAuthentication': '0',
                'trafficDataStorage': '0',
                'trafficDatabase': '0',
                'trafficGaming': '0',
                'trafficMAIL': '0',
                'trafficManagement': '46',
                'trafficNetworkOperation': '6552399',
                'trafficNetworkTunnel': '82980',
                'trafficP2P': '0',
                'trafficPrinting': '0',
                'trafficSocial': '0',
                'trafficSystemsOperation': '12802',
                'trafficUnclassified': '4420',
                'trafficVPN': '0',
                'trafficVoiceVideo': '0',
                'trafficWEB': '0'}},
 {'exporter1': {'exporterAddress': 'EOL'}}]
```

def GetNetflowTemplates(user, password, server, port): This function returns all the received netflow templates from Netflow9 and IPFIX exporters.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'netflowTemplate0': {'HasNBARFields': 'FALSE',
                       'IsOptionRecord': 'FALSE',
                       'exporterAddress': '10.100.1.1',
                       'fieldCode0': '62',
                       'fieldCode1': '27',
                       'fieldCode10': '54',
                       'fieldCode11': '243',
                       'fieldCode12': '254',
                       'fieldCode13': '56',
                       'fieldCode14': '81',
                       'fieldCode15': '80',
                       'fieldCode16': '57',
                       'fieldCode17': '128',
                       'fieldCode18': '7',
                       'fieldCode19': '11',
                       'fieldCode2': '28',
                       'fieldCode20': '182',
                       'fieldCode21': '183',
                       'fieldCode22': '180',
                       'fieldCode23': '181',
                       'fieldCode24': '224',
                       'fieldCode25': '89',
                       'fieldCode26': '31',
                       'fieldCode27': '176',
                       'fieldCode28': '177',
                       'fieldCode29': '178',
                       'fieldCode3': '15',
                       'fieldCode30': '179',
                       'fieldCode31': '61',
                       'fieldCode32': '60',
                       'fieldCode33': '5',
                       'fieldCode34': '195',
                       'fieldCode35': '196',
                       'fieldCode36': '4',
                       'fieldCode37': '192',
                       'fieldCode38': '197',
                       'fieldCode39': '16',
                       'fieldCode4': '8',
                       'fieldCode40': '1',
                       'fieldCode41': '2',
                       'fieldCode42': '22',
                       'fieldCode43': '21',
                       'fieldCode44': '10',
                       'fieldCode45': '14',
                       'fieldCode46': '152',
                       'fieldCode47': '153',
                       'fieldCode5': '12',
                       'fieldCode6': '208',
                       'fieldCode7': '184',
                       'fieldCode8': '48',
                       'fieldCode9': '95',
                       'fieldName0': 'ipNextHopIPv6Address',
                       'fieldName1': 'sourceIPv6Address',
                       'fieldName10': 'fragmentIdentification',
                       'fieldName11': 'dot1qVlanId',
                       'fieldName12': 'postDot1qVlanID',
                       'fieldName13': 'sourceMacAddress',
                       'fieldName14': 'postSourceMacAddress',
                       'fieldName15': 'destinationMacAddress',
                       'fieldName16': 'postDestinationMacAddress',
                       'fieldName17': 'bgpNextAdjacentAsNumber',
                       'fieldName18': 'sourceTransportPort',
                       'fieldName19': 'destinationTransportPort',
                       'fieldName2': 'destinationIPv6Address',
                       'fieldName20': 'tcpSourcePort',
                       'fieldName21': 'tcpDestinationPort',
                       'fieldName22': 'udpSourcePort',
                       'fieldName23': 'udpDestinationPort',
                       'fieldName24': 'ipTotalLength',
                       'fieldName25': 'forwardingStatus',
                       'fieldName26': 'flowLabelIPv6',
                       'fieldName27': 'cisco_asa_NF_F_ICMP_TYPE',
                       'fieldName28': 'cisco_asa_NF_F_ICMP_CODE',
                       'fieldName29': 'cisco_asa_NF_F_ICMP_TYPE_IPV6',
                       'fieldName3': 'ipNextHopIPv4Address',
                       'fieldName30': 'cisco_asa_NF_F_ICMP_CODE_IPV6',
                       'fieldName31': 'flowDirection',
                       'fieldName32': 'ipVersion',
                       'fieldName33': 'ipClassOfService',
                       'fieldName34': 'ipDiffServCodePoint',
                       'fieldName35': 'ipPrecedence',
                       'fieldName36': 'protocolIdentifier',
                       'fieldName37': 'ipTTL',
                       'fieldName38': 'fragmentFlags',
                       'fieldName39': 'bgpSourceAsNumber',
                       'fieldName4': 'sourceIPv4Address',
                       'fieldName40': 'byteDeltaCount',
                       'fieldName41': 'packetDeltaCount',
                       'fieldName42': 'flowStartSysUpTime',
                       'fieldName43': 'flowEndSysUpTime',
                       'fieldName44': 'ingressInterface',
                       'fieldName45': 'egressInterface',
                       'fieldName46': 'flowStartMilliseconds',
                       'fieldName47': 'flowEndMilliseconds',
                       'fieldName5': 'destinationIPv4Address',
                       'fieldName6': 'ipv4Options',
                       'fieldName7': 'tcpSequenceNumber',
                       'fieldName8': 'flowSamplerID',
                       'fieldName9': 'applicationId',
                       'fieldSize0': '16',
                       'fieldSize1': '16',
                       'fieldSize10': '4',
                       'fieldSize11': '2',
                       'fieldSize12': '2',
                       'fieldSize13': '6',
                       'fieldSize14': '6',
                       'fieldSize15': '6',
                       'fieldSize16': '6',
                       'fieldSize17': '2',
                       'fieldSize18': '2',
                       'fieldSize19': '2',
                       'fieldSize2': '16',
                       'fieldSize20': '2',
                       'fieldSize21': '2',
                       'fieldSize22': '2',
                       'fieldSize23': '2',
                       'fieldSize24': '2',
                       'fieldSize25': '1',
                       'fieldSize26': '3',
                       'fieldSize27': '1',
                       'fieldSize28': '1',
                       'fieldSize29': '1',
                       'fieldSize3': '4',
                       'fieldSize30': '1',
                       'fieldSize31': '1',
                       'fieldSize32': '1',
                       'fieldSize33': '1',
                       'fieldSize34': '1',
                       'fieldSize35': '1',
                       'fieldSize36': '1',
                       'fieldSize37': '1',
                       'fieldSize38': '1',
                       'fieldSize39': '2',
                                               'fieldSize4': '4',
                                               'fieldSize40': '4',
                                               'fieldSize41': '4',
                                               'fieldSize42': '4',
                                               'fieldSize43': '4',
                                               'fieldSize44': '4',
                                               'fieldSize45': '4',
                                               'fieldSize46': '8',
                                               'fieldSize47': '8',
                                               'fieldSize5': '4',
                                               'fieldSize6': '4',
                                               'fieldSize7': '4',
                                               'fieldSize8': '4',
                                               'fieldSize9': '4',
                                               'fieldType0': 'IPv6 Address',
                                               'fieldType1': 'IPv6 Address',
                                               'fieldType10': 'Numeric 2 byte',
                                               'fieldType11': 'Numeric 2 byte',
                                               'fieldType12': 'Numeric 2 byte',
                                               'fieldType13': 'MAC Address',
                                               'fieldType14': 'MAC Address',
                                               'fieldType15': 'MAC Address',
                                               'fieldType16': 'MAC Address',
                                               'fieldType17': 'Numeric 2 byte',
                                               'fieldType18': 'Numeric 2 byte',
                                               'fieldType19': 'Numeric 2 byte',
                                               'fieldType2': 'IPv6 Address',
                                               'fieldType20': 'Numeric 2 byte',
                                               'fieldType21': 'Numeric 2 byte',
                                               'fieldType22': 'Numeric 2 byte',
                                               'fieldType23': 'Numeric 2 byte',
                                               'fieldType24': 'Numeric 8 byte',
                                               'fieldType25': 'Numeric 1 byte',
                                               'fieldType26': 'Numeric 4 byte',
                                               'fieldType27': 'Numeric 1 byte',
                                               'fieldType28': 'Numeric 1 byte',
                                               'fieldType29': 'Numeric 1 byte',
                                               'fieldType3': 'IPv4 Address',
                                               'fieldType30': 'Numeric 1 byte',
                                               'fieldType31': 'Numeric 1 byte',
                                               'fieldType32': 'Numeric 1 byte',
                                               'fieldType33': 'Numeric 1 byte',
                                               'fieldType34': 'Numeric 1 byte',
                                               'fieldType35': 'Numeric 1 byte',
                                               'fieldType36': 'Numeric 1 byte',
                                               'fieldType37': 'Numeric 1 byte',
                                               'fieldType38': 'Numeric 1 byte',
                                               'fieldType39': 'Numeric 2 byte',
                                               'fieldType4': 'IPv4 Address',
                                               'fieldType40': 'Numeric 4 byte',
                                               'fieldType41': 'Numeric 4 byte',
                                               'fieldType42': 'Numeric 4 byte',
                                               'fieldType43': 'Numeric 4 byte',
                                               'fieldType44': 'Numeric 2 byte',
                                               'fieldType45': 'Numeric 2 byte',
                                               'fieldType46': 'Numeric 8 byte',
                                               'fieldType47': 'Numeric 8 byte',
                                               'fieldType5': 'IPv4 Address',
                                               'fieldType6': 'Numeric 4 byte',
                                               'fieldType7': 'Numeric 4 byte',
                                               'fieldType8': 'Numeric 1 byte',
                                               'fieldType9': 'Numeric 4 byte',
                                               'hits': '77503',
                                               'lastUpdate': '20/03/2017 12:03:07',
                                               'netflowVersion': '10',
                                               'numberOfFields': '48',
                                               'templateID': '259'}},
                       < … OMISSIS … >
                        {'netflowTemplate53': {'HasNBARFields': 'FALSE',
                                                'IsOptionRecord': 'FALSE',
                                                'exporterAddress': '10.1.30.150',
                                                'fieldCode0': '60',
                                                'fieldCode1': '27',
                                                'fieldCode10': '5',
                                                'fieldCode11': '7',
                                                'fieldCode12': '11',
                                                'fieldCode13': '31',
                                                'fieldCode14': '64',
                       'fieldCode15': '21',
                       'fieldCode16': '22',
                       'fieldCode17': '1',
                       'fieldCode18': '2',
                       'fieldCode19': '80',
                       'fieldCode2': '29',
                       'fieldCode20': '81',
                       'fieldCode3': '10',
                       'fieldCode4': '28',
                       'fieldCode5': '30',
                       'fieldCode6': '14',
                       'fieldCode7': '62',
                       'fieldCode8': '4',
                       'fieldCode9': '6',
                       'fieldName0': 'ipVersion',
                       'fieldName1': 'sourceIPv6Address',
                       'fieldName10': 'ipClassOfService',
                       'fieldName11': 'sourceTransportPort',
                       'fieldName12': 'destinationTransportPort',
                       'fieldName13': 'flowLabelIPv6',
                       'fieldName14': 'ipv6ExtensionHeaders',
                       'fieldName15': 'flowEndSysUpTime',
                       'fieldName16': 'flowStartSysUpTime',
                       'fieldName17': 'byteDeltaCount',
                       'fieldName18': 'packetDeltaCount',
                       'fieldName19': 'destinationMacAddress',
                       'fieldName2': 'sourceIPv6PrefixLength',
                       'fieldName20': 'postSourceMacAddress',
                       'fieldName3': 'ingressInterface',
                       'fieldName4': 'destinationIPv6Address',
                       'fieldName5': 'destinationIPv6PrefixLength',
                       'fieldName6': 'egressInterface',
                       'fieldName7': 'ipNextHopIPv6Address',
                       'fieldName8': 'protocolIdentifier',
                       'fieldName9': 'tcpControlBits',
                       'fieldSize0': '1',
                       'fieldSize1': '16',
                       'fieldSize10': '1',
                       'fieldSize11': '2',
                       'fieldSize12': '2',
                       'fieldSize13': '4',
                       'fieldSize14': '4',
                       'fieldSize15': '4',
                       'fieldSize16': '4',
                       'fieldSize17': '4',
                       'fieldSize18': '4',
                       'fieldSize19': '6',
                       'fieldSize2': '1',
                       'fieldSize20': '6',
                       'fieldSize3': '4',
                       'fieldSize4': '16',
                       'fieldSize5': '1',
                       'fieldSize6': '4',
                       'fieldSize7': '16',
                       'fieldSize8': '1',
                       'fieldSize9': '1',
                       'fieldType0': 'Numeric 1 byte',
                       'fieldType1': 'IPv6 Address',
                       'fieldType10': 'Numeric 1 byte',
                       'fieldType11': 'Numeric 2 byte',
                       'fieldType12': 'Numeric 2 byte',
                       'fieldType13': 'Numeric 4 byte',
                       'fieldType14': 'Numeric 4 byte',
                       'fieldType15': 'Numeric 4 byte',
                       'fieldType16': 'Numeric 4 byte',
                       'fieldType17': 'Numeric 4 byte',
                       'fieldType18': 'Numeric 4 byte',
                       'fieldType19': 'MAC Address',
                       'fieldType2': 'Numeric 1 byte',
                       'fieldType20': 'MAC Address',
                       'fieldType3': 'Numeric 2 byte',
                       'fieldType4': 'IPv6 Address',
                       'fieldType5': 'Numeric 1 byte',
                       'fieldType6': 'Numeric 2 byte',
                       'fieldType7': 'IPv6 Address',
                       'fieldType8': 'Numeric 1 byte',
                       'fieldType9': 'Numeric 1 byte',
                       'hits': '715',
                                       'lastUpdate': '20/03/2017 11:08:24',
                                       'netflowVersion': '9',
                                       'numberOfFields': '21',
                                       'templateID': '257'}}]
```

def GetNBARTemplates(user, password, server, port): This function returns all the received netflow templates containing NBAR information from Netflow9 and IPFIX exporters.

►This feature is still experimental. Could be subject of change.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'NBAR_NetflowTemplate0': {'NBAR_Appl_Category_app_0': '',
                            'NBAR_Appl_Category_app_1': '',
                            'NBAR_Appl_Category_app_10': '',
                            'NBAR_Appl_Category_app_100': '',
                            'NBAR_Appl_Category_app_1000': '',
                            'NBAR_Appl_Category_app_1001': '',
                            'NBAR_Appl_Category_app_1002': '',
                            'NBAR_Appl_Category_app_1003': '',
                            'NBAR_Appl_Description_app_0': 'DEPRECATED, '
                                                           'traffic will '
                                                           'not match',
                            'NBAR_Appl_Description_app_1': 'Internet '
                                                           'Control '
                                                           'Message '
                                                           'Protocol',
                            'NBAR_Appl_Description_app_10': 'PUP',
                            'NBAR_Appl_Description_app_100': 'IPX in IP',
                            'NBAR_Appl_Description_app_1000': 'Online '
                                                              'Music '
                                                              'Service',
                            'NBAR_Appl_Description_app_1001': 'Microsoft '
                                                              'Activesync '
                                                              'protocol,mobile '
                                                              'data '
                                                              'synchronizati',
                            'NBAR_Appl_Description_app_1002': 'Real Time '
                                                              'Messaging '
                                                              'Protocol '
                                                              'tunneled '
                            'NBAR_Appl_Description_app_1003': 'Cisco '
                                                              'Digital '
                                                              'Media Player',
                                                            'protocol',
                            'NBAR_Appl_Group_app_0': '',
                            'NBAR_Appl_Group_app_1': '',
                            'NBAR_Appl_Group_app_10': '',
                            'NBAR_Appl_Group_app_100': '',
                            'NBAR_Appl_Group_app_1000': '',
                            'NBAR_Appl_Group_app_1001': '',
                            'NBAR_Appl_Group_app_1002': '',
                            'NBAR_Appl_Group_app_1003': '',
                            'NBAR_Appl_ID_app_0': '1:0',
                            'NBAR_Appl_ID_app_1': '1:1',
                            'NBAR_Appl_ID_app_10': '1:12',
                            'NBAR_Appl_ID_app_100': '1:111',
                            'NBAR_Appl_ID_app_1000': '13:489',
                            'NBAR_Appl_ID_app_1001': '13:490',
                            'NBAR_Appl_ID_app_1002': '13:491',
                            'NBAR_Appl_ID_app_1003': '13:492',
                            'NBAR_Appl_IsEncrypted_app_0': '',
                            'NBAR_Appl_IsEncrypted_app_1': '',
                            'NBAR_Appl_IsEncrypted_app_10': '',
                            'NBAR_Appl_IsEncrypted_app_100': '',
                            'NBAR_Appl_IsEncrypted_app_1000': '',
                            'NBAR_Appl_IsEncrypted_app_1001': '',
                            'NBAR_Appl_IsEncrypted_app_1002': '',
                            'NBAR_Appl_IsEncrypted_app_1003': '',
                            'NBAR_Appl_IsP2P_app_0': '',
                            'NBAR_Appl_IsP2P_app_1': '',
                            'NBAR_Appl_IsP2P_app_10': '',
                            'NBAR_Appl_IsP2P_app_100': '',
                            'NBAR_Appl_IsP2P_app_1000': '',
                            'NBAR_Appl_IsP2P_app_1001': '',
                            'NBAR_Appl_IsP2P_app_1002': '',
                            'NBAR_Appl_IsP2P_app_1003': '',
                            'NBAR_Appl_IsTunnel_app_0': '',
                            'NBAR_Appl_IsTunnel_app_1': '',
                            'NBAR_Appl_IsTunnel_app_10': '',
'NBAR_Appl_IsTunnel_app_100': '',
'NBAR_Appl_IsTunnel_app_1000': '',
'NBAR_Appl_IsTunnel_app_1001': '',
'NBAR_Appl_IsTunnel_app_1002': '',
'NBAR_Appl_IsTunnel_app_1003': '',
'NBAR_Appl_Name_app_0': 'hopopt',
'NBAR_Appl_Name_app_1': 'icmp',
'NBAR_Appl_Name_app_10': 'pup',
'NBAR_Appl_Name_app_100': 'ipx-in-ip',
'NBAR_Appl_Name_app_1000': 'rhapsody',
'NBAR_Appl_Name_app_1001': 'activesync',
'NBAR_Appl_Name_app_1002': 'rtmpt',
'NBAR_Appl_Name_app_1003': 'dmp',
'NBAR_Appl_SubCategory_app_0': '',
'NBAR_Appl_SubCategory_app_1': '',
'NBAR_Appl_SubCategory_app_10': '',
'NBAR_Appl_SubCategory_app_100': '',
'NBAR_Appl_SubCategory_app_1000': '',
'NBAR_Appl_SubCategory_app_1001': '',
'NBAR_Appl_SubCategory_app_1002': '',
'NBAR_Appl_SubCategory_app_1003': '',
'NBAR_Bytes_app_0': '0',
'NBAR_Bytes_app_1': '0',
'NBAR_Bytes_app_10': '0',
'NBAR_Bytes_app_100': '0',
'NBAR_Bytes_app_1000': '0',
'NBAR_Bytes_app_1001': '0',
'NBAR_Bytes_app_1002': '0',
'NBAR_Bytes_app_1003': '0',
'NBAR_Hits_app_0': '0',
'NBAR_Hits_app_1': '0',
'NBAR_Hits_app_10': '0',
'NBAR_Hits_app_100': '0',
'NBAR_Hits_app_1000': '0',
'NBAR_Hits_app_1001': '0',
'NBAR_Hits_app_1002': '0',
'NBAR_Hits_app_1003': '0',
'NBAR_LastUpdate': '20/03/2017 11:54:48',
'NBAR_Packets_app_0': '0',
'NBAR_Packets_app_1': '0',
'NBAR_Packets_app_10': '0',
'NBAR_Packets_app_100': '0',
'NBAR_Packets_app_1000': '0',
'NBAR_Packets_app_1001': '0',
'NBAR_Packets_app_1002': '0',
'NBAR_Packets_app_1003': '0',
'NBAR_TemplateID': '257',
'NBAR_TemplateNetflowVersion': '10',
'NBAR_TemplateNumberOfFields': '3',
'NBAR_exporterAddress': '10.100.1.1',
'entries': '1025'}}]
```

def GetSingleExporter(user, password, server, port,ip): Gets Single Exporter Information from the Fl0wer server.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'FlowExporter_single_entry': {'ICMPTrafficBytes': '29300229',
                                'ICMPTrafficPackets': '154466',
                                'IGMPTrafficBytes': '0',
                                'IGMPTrafficPackets': '0',
                                'IPv4TrafficBytes': '243930271',
                                'IPv4TrafficPackets': '544830',
                                'IPv6TrafficBytes': '176782',
                                'IPv6TrafficPackets': '2903',
                                'OtherTrafficBytes': '176782',
                                'OtherTrafficPackets': '2903',
                                'TCPTrafficBytes': '112632046',
                                'TCPTrafficFlows': '183850',
                                'TCPTrafficPackets': '184007',
                                'UDPTrafficBytes': '101997996',
                                'UDPTrafficFlows': '194689',
                                'UDPTrafficPackets': '101997996',
                                'avgFlowSec': '12',
                                'avgPktSec': '1',
                                'counter_bytes_FLOW_HOST_INTERNAL': '0',
                                'counter_bytes_FLOW_INTERNAL_TO_INTERNAL': '117651428',
                                'counter_bytes_FLOW_INTERNAL_TO_INTERNET': '16677619',
                                'counter_bytes_FLOW_INTERNAL_TO_UNKNOWN': '0',
                                'counter_bytes_FLOW_INTERNET_TO_INTERNAL': '109778006',
                                'counter_bytes_FLOW_INTERNET_TO_INTERNET': '0',
                                'counter_bytes_FLOW_INTERNET_TO_UNKNOWN': '0',
                                'counter_bytes_FLOW_UNKNOWN': '0',
                                'counter_packets_FLOW_HOST_INTERNAL': '0',
                                'counter_packets_FLOW_INTERNAL_TO_INTERNAL': '0',
                                'counter_packets_FLOW_INTERNAL_TO_INTERNET': '127685',
                                'counter_packets_FLOW_INTERNAL_TO_UNKNOWN': '0',
                                'counter_packets_FLOW_INTERNET_TO_INTERNAL': '143582',
                                'counter_packets_FLOW_INTERNET_TO_INTERNET': '0',
                                'counter_packets_FLOW_INTERNET_TO_UNKNOWN': '0',
                                'counter_packets_FLOW_UNKNOWN': '0',
                                'exporterAddress': '10.100.1.1',
                                'firstPacketTime': '20/03/2017 00:01:13',
                                'lastPacketTime': '20/03/2017 12:09:41',
                                'nf10flows': '528568',
                                'nf10packets': '85015',
                                'nf10templates': '1462',
                                'nf1flows': '0',
                                'nf1packets': '0',
                                'nf5flows': '0',
                                'nf5packets': '0',
                                'nf9flows': '0',
                                'nf9packets': '0',
                                'nf9templates': '0',
                                'tcp_ACK_count': '0',
                                'tcp_CWR_count': '0',
                                'tcp_ECN_count': '0',
                                'tcp_FIN_count': '0',
                                'tcp_PSH_count': '0',
                                'tcp_RST_count': '0',
                                'tcp_SYN_count': '0',
                                'tcp_URG_count': '0',
                                'totBytesSeen': '244107053',
                                'totPacketsSeen': '547733',
                                'trafficAuthentication': '0',
                                'trafficDataStorage': '0',
                                'trafficDatabase': '0',
                                'trafficGaming': '1519',
                                'trafficMAIL': '1656138',
                                'trafficManagement': '89943627',
                                'trafficNetworkOperation': '32872381',
                                'trafficNetworkTunnel': '404884',
                                'trafficP2P': '1300497',
                                'trafficPrinting': '0',
                                'trafficSocial': '8161536',
                                'trafficSystemsOperation': '4437',
                                'trafficUnclassified': '0',
                                'trafficVPN': '0',
'trafficVoiceVideo': '467',
'trafficWEB': '109761567'}}]
```

Flow functions

def GetFlowsAll(user, password, server, port): This functions returns the last N flows where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
            'FlowDuration': '0',
            'FlowID': '1490011360796773',
            'NBAR': '',
            'NPAR': 'UDP 53 DNS Protocol',
            'RULE': '',
            'XINFO': '10.1.30.150 is an allowed internal DNS server '
                     'replying to 10.1.30.220',
            'connBytes': '117',
            'connPackets': '1',
            'dateReceived': '20/03/2017 12:10:26',
            'dstAddr': '10.1.30.220',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': '',
            'dstOrganization': '',
            'dstPort': '24643',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.1.30.220',
            'firstPacketTime': '20/03/2017 12:10:10',
            'ipProtocol': 'UDP',
            'ipVersion': '4',
            'lastPacketTime': '20/03/2017 12:10:10',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '10',
            'nextHop': '0.0.0.0',
            'srcAddr': '10.1.30.150',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': '00:00:5e:00:01:96',
            'srcOrganization': '',
            'srcPort': '53',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '0',
            'templateID': '258',
            'trafficCategory': 'NETWORK OPERATION',
            'trafficClassification': '[NORMAL] '}},
< … OMISSIS … >
 {'Flow1999': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
               'FlowDuration': '0',
               'FlowID': '1490011300285550',
               'NBAR': 'AppID: 3741384717 (13:479) was not found - NBAR '
                       'data still not available',
               'NPAR': 'ICMP Protocol Code: ECHO REPLY',
               'RULE': '',
               'XINFO': '',
               'connBytes': '56',
               'connPackets': '1',
               'dateReceived': '20/03/2017 12:12:00',
               'dstAddr': '10.100.0.1',
               'dstAreaCode': '0',
               'dstAssignedNetmask': '0',
               'dstBeginIP': '',
               'dstCity': '',
               'dstCountryCode': '',
               'dstEndIP': '',
               'dstHostname': '',
               'dstISP_AS': '',
               'dstLatitude': '0.000000',
               'dstLongitude': '0.000000',
               'dstMACAddress': '00:0d:b9:34:16:18',
               'dstOrganization': '',
               'dstPort': '0',
               'dstPostalCode': '',
               'dstRegion': '',
               'dstRegionName': '',
               'dstTimezone': '',
               'dstVLAN': '0',
               'exporterAddress': '10.100.1.1',
               'firstPacketTime': '20/03/2017 12:11:45',
               'ipProtocol': 'ICMP',
               'ipVersion': '4',
               'lastPacketTime': '20/03/2017 12:11:45',
               'natInsideDstAddr': '',
               'natInsideSrcAddr': '',
               'netflowVersion': '10',
               'nextHop': '10.100.1.20',
               'srcAddr': '10.100.3.21',
               'srcAreaCode': '0',
               'srcAssignedNetmask': '0',
               'srcBeginIP': '',
               'srcCity': '',
               'srcCountryCode': '',
               'srcEndIP': '',
               'srcHostname': '',
               'srcISP_AS': '',
               'srcLatitude': '0.000000',
               'srcLongitude': '0.000000',
               'srcMACAddress': 'd4:ca:6d:0d:ba:e4',
               'srcOrganization': '',
               'srcPort': '0',
               'srcPostalCode': '',
               'srcRegion': '',
               'srcRegionName': '',
               'srcTimezone': '',
               'srcVLAN': '0',
               'tcp_flags': '0',
               'templateID': '259',
               'trafficCategory': 'NETWORK OPERATION',
               'trafficClassification': '[UNRATED] '}},
 {'Flow2000': {'exporterAddress': 'EOL'}}]
```

def GetTCPv4Flows(user, password, server, port): This functions returns the last N TCP IPv4 flows where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
            'FlowDuration': '0',
            'FlowID': '1490009510390981',
            'NBAR': '',
            'NPAR': 'TCP 6080 Ubiquiti mFI Server',
            'RULE': '',
            'XINFO': '',
            'connBytes': '80',
            'connPackets': '2',
            'dateReceived': '20/03/2017 12:12:19',
            'dstAddr': '10.1.60.130',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': 'd4:ca:6d:d8:3f:d3',
            'dstOrganization': '',
            'dstPort': '39446',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.1.30.100',
            'firstPacketTime': '20/03/2017 12:12:19',
            'ipProtocol': 'TCP',
            'ipVersion': '4',
            'lastPacketTime': '20/03/2017 12:12:19',
            'natInsideDstAddr': '10.1.60.130',
            'natInsideSrcAddr': '10.1.30.220',
            'netflowVersion': '10',
            'nextHop': '0.0.0.0',
            'srcAddr': '10.1.30.220',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': 'd4:ca:6d:d8:3f:d2',
            'srcOrganization': '',
            'srcPort': '6080',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '20',
            'templateID': '258',
            'trafficCategory': 'MANAGEMENT',
            'trafficClassification': '[UNRATED] '}},
< … OMISSIS … >
 {'Flow235': {'FlowDirection': 'INTERNAL_TO_INTERNET',
              'FlowDuration': '0',
              'FlowID': '1490010240171200',
              'NBAR': '',
              'NPAR': 'TCP 993 MAIL IMAPv4/IMAPS/IMAP-SSL Protocol',
              'RULE': '',
              'XINFO': '',
              'connBytes': '52',
              'connPackets': '1',
              'dateReceived': '20/03/2017 12:14:06',
              'dstAddr': '81.88.58.204',
              'dstAreaCode': '0',
              'dstAssignedNetmask': '0',
              'dstBeginIP': '',
              'dstCity': '',
              'dstCountryCode': '',
              'dstEndIP': '',
              'dstHostname': '',
              'dstISP_AS': '',
              'dstLatitude': '0.000000',
              'dstLongitude': '0.000000',
              'dstMACAddress': '',
              'dstOrganization': '',
              'dstPort': '993',
              'dstPostalCode': '',
              'dstRegion': '',
              'dstRegionName': '',
              'dstTimezone': '',
              'dstVLAN': '0',
              'exporterAddress': '10.100.0.20',
              'firstPacketTime': '20/03/2017 12:14:06',
              'ipProtocol': 'TCP',
              'ipVersion': '4',
              'lastPacketTime': '20/03/2017 12:14:06',
              'natInsideDstAddr': '',
              'natInsideSrcAddr': '',
              'netflowVersion': '9',
              'nextHop': '10.100.1.1',
              'srcAddr': '10.1.60.194',
              'srcAreaCode': '0',
              'srcAssignedNetmask': '0',
              'srcBeginIP': '',
              'srcCity': '',
              'srcCountryCode': '',
              'srcEndIP': '',
              'srcHostname': '',
              'srcISP_AS': '',
              'srcLatitude': '0.000000',
              'srcLongitude': '0.000000',
              'srcMACAddress': '',
              'srcOrganization': '',
              'srcPort': '59062',
              'srcPostalCode': '',
              'srcRegion': '',
              'srcRegionName': '',
              'srcTimezone': '',
              'srcVLAN': '0',
              'tcp_flags': '16',
              'templateID': '256',
              'trafficCategory': 'MAIL',
              'trafficClassification': '[UNRATED] '}},
 {'Flow236': {'exporterAddress': 'EOL'}}]
```

def GetUDPv4Flows(user, password, server, port): This functions returns the last N UDP IPv4 flows where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
            'FlowDuration': '0',
            'FlowID': '1490010713395308',
            'NBAR': '',
            'NPAR': 'UDP 2056 Cisco Netflow/IPFIX Protocol',
            'RULE': '',
            'XINFO': '',
            'connBytes': '108',
            'connPackets': '1',
            'dateReceived': '20/03/2017 12:16:37',
            'dstAddr': '10.1.30.210',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': '',
            'dstOrganization': '',
            'dstPort': '2056',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.1.30.220',
            'firstPacketTime': '20/03/2017 12:16:22',
            'ipProtocol': 'UDP',
            'ipVersion': '4',
            'lastPacketTime': '20/03/2017 12:16:22',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '10',
            'nextHop': '0.0.0.0',
            'srcAddr': '10.100.4.20',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': '00:00:5e:00:01:1e',
            'srcOrganization': '',
            'srcPort': '53638',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '0',
            'templateID': '258',
            'trafficCategory': 'MANAGEMENT',
            'trafficClassification': '[UNRATED] '}},
< … OMISSIS … >
 {'Flow1591': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
               'FlowDuration': '0',
               'FlowID': '1490011718886754',
               'NBAR': '',
               'NPAR': 'UDP 53 DNS Protocol',
               'RULE': '',
              'XINFO': '10.1.30.150 is an allowed internal DNS server '
                       'queried by 10.1.30.220',
              'connBytes': '140',
              'connPackets': '2',
              'dateReceived': '20/03/2017 12:16:13',
              'dstAddr': '10.1.30.150',
              'dstAreaCode': '0',
              'dstAssignedNetmask': '0',
              'dstBeginIP': '',
              'dstCity': '',
              'dstCountryCode': '',
              'dstEndIP': '',
              'dstHostname': '',
              'dstISP_AS': '',
              'dstLatitude': '0.000000',
              'dstLongitude': '0.000000',
              'dstMACAddress': '00:00:00:00:00:00',
              'dstOrganization': '',
              'dstPort': '53',
              'dstPostalCode': '',
              'dstRegion': '',
              'dstRegionName': '',
              'dstTimezone': '',
              'dstVLAN': '0',
              'exporterAddress': '10.1.30.100',
              'firstPacketTime': '20/03/2017 12:16:13',
              'ipProtocol': 'UDP',
              'ipVersion': '4',
              'lastPacketTime': '20/03/2017 12:16:13',
              'natInsideDstAddr': '10.1.30.150',
              'natInsideSrcAddr': '10.1.30.220',
              'netflowVersion': '10',
              'nextHop': '10.1.30.150',
              'srcAddr': '10.1.30.220',
              'srcAreaCode': '0',
              'srcAssignedNetmask': '0',
              'srcBeginIP': '',
              'srcCity': '',
              'srcCountryCode': '',
              'srcEndIP': '',
              'srcHostname': '',
              'srcISP_AS': '',
              'srcLatitude': '0.000000',
              'srcLongitude': '0.000000',
              'srcMACAddress': 'd4:ca:6d:d8:3f:d2',
              'srcOrganization': '',
              'srcPort': '22014',
              'srcPostalCode': '',
              'srcRegion': '',
              'srcRegionName': '',
              'srcTimezone': '',
              'srcVLAN': '0',
              'tcp_flags': '0',
              'templateID': '258',
              'trafficCategory': 'NETWORK OPERATION',
              'trafficClassification': '[NORMAL] '}},
{'Flow1592': {'exporterAddress': 'EOL'}}]
```

def GetTCPv6Flows(user, password, server, port): This functions returns the last N TCP IPv6 flows where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNET',
            'FlowDuration': '0',
            'FlowID': '1490031588037941',
            'NBAR': '',
            'NPAR': 'Hurricane IPv6 Tunnel Broker',
            'RULE': '',
            'XINFO': '',
            'connBytes': '120',
            'connPackets': '2',
            'dateReceived': '20/03/2017 18:16:14',
            'dstAddr': '2a03:2880:f112:83:face:b00c:0:25de',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': '',
            'dstOrganization': '',
            'dstPort': '443',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.100.0.20',
            'firstPacketTime': '20/03/2017 18:16:14',
            'ipProtocol': 'TCP',
            'ipVersion': '6',
            'lastPacketTime': '20/03/2017 18:16:14',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '9',
            'nextHop': '2001:470:8b5b:0:10:100:1:1',
            'srcAddr': '2001:470:8b5b:0:10:1:30:143',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': '',
            'srcOrganization': '',
            'srcPort': '50490',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '4',
            'templateID': '259',
            'trafficCategory': 'NETWORK TUNNEL',
            'trafficClassification': '[UNRATED] '}},
< … OMISSIS … >
 {'Flow10': {'FlowDirection': 'INTERNAL_TO_INTERNET',
            'FlowDuration': '0',
            'FlowID': '1490031588952850',
            'NBAR': '',
            'NPAR': 'Hurricane IPv6 Tunnel Broker',
            'RULE': '',
           'XINFO': '',
           'connBytes': '120',
           'connPackets': '2',
           'dateReceived': '20/03/2017 18:16:15',
           'dstAddr': '2a03:2880:f112:83:face:b00c:0:25de',
           'dstAreaCode': '0',
           'dstAssignedNetmask': '0',
           'dstBeginIP': '',
           'dstCity': '',
           'dstCountryCode': '',
           'dstEndIP': '',
           'dstHostname': '',
           'dstISP_AS': '',
           'dstLatitude': '0.000000',
           'dstLongitude': '0.000000',
           'dstMACAddress': '',
           'dstOrganization': '',
           'dstPort': '443',
           'dstPostalCode': '',
           'dstRegion': '',
           'dstRegionName': '',
           'dstTimezone': '',
           'dstVLAN': '0',
           'exporterAddress': '10.100.0.20',
           'firstPacketTime': '20/03/2017 18:16:15',
           'ipProtocol': 'TCP',
           'ipVersion': '6',
           'lastPacketTime': '20/03/2017 18:16:15',
           'natInsideDstAddr': '',
           'natInsideSrcAddr': '',
           'netflowVersion': '9',
           'nextHop': '2001:470:8b5b:0:10:100:1:1',
           'srcAddr': '2001:470:8b5b:0:10:1:30:143',
           'srcAreaCode': '0',
           'srcAssignedNetmask': '0',
           'srcBeginIP': '',
           'srcCity': '',
           'srcCountryCode': '',
           'srcEndIP': '',
           'srcHostname': '',
           'srcISP_AS': '',
           'srcLatitude': '0.000000',
           'srcLongitude': '0.000000',
           'srcMACAddress': '',
           'srcOrganization': '',
           'srcPort': '50490',
           'srcPostalCode': '',
           'srcRegion': '',
           'srcRegionName': '',
           'srcTimezone': '',
           'srcVLAN': '0',
           'tcp_flags': '4',
           'templateID': '259',
           'trafficCategory': 'NETWORK TUNNEL',
           'trafficClassification': '[UNRATED] '}},
{'Flow2': {'exporterAddress': 'EOL'}}]
```

def GetUDPv6Flows(user, password, server, port): This functions returns the last N UDP IPv6 flows where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNET',
            'FlowDuration': '0',
            'FlowID': '1490009119302268',
            'NBAR': '',
            'NPAR': 'UDP 5678 Mikrotik Discovery Protocol/Apple Server '
                    'SNATMAP iChat',
            'RULE': '',
            'XINFO': '',
            'connBytes': '320',
            'connPackets': '2',
            'dateReceived': '20/03/2017 12:17:23',
            'dstAddr': 'ff02::1',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': '00:00:00:00:00:00',
            'dstOrganization': '',
            'dstPort': '5678',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.1.30.20',
            'firstPacketTime': '20/03/2017 12:17:23',
            'ipProtocol': 'UDP',
            'ipVersion': '6',
            'lastPacketTime': '20/03/2017 12:17:23',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '9',
            'nextHop': 'ff02::1',
            'srcAddr': 'fe80::d6ca:6dff:fed8:3f11',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': '00:0c:42:e9:25:40',
            'srcOrganization': '',
            'srcPort': '5678',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '0',
            'templateID': '257',
            'trafficCategory': 'NETWORK OPERATION',
            'trafficClassification': '[UNRATED] '}},
< … OMISSIS … >
 {'Flow10': {'FlowDirection': 'INTERNAL_TO_INTERNET',
             'FlowDuration': '0',
             'FlowID': '1490010533668907',
             'NBAR': '',
             'NPAR': 'UDP 5678 Mikrotik Discovery Protocol/Apple Server '
```

```
                'SNATMAP iChat',
        'RULE': '',
        'XINFO': '',
        'connBytes': '362',
        'connPackets': '2',
        'dateReceived': '20/03/2017 12:18:23',
        'dstAddr': 'ff02::1',
        'dstAreaCode': '0',
        'dstAssignedNetmask': '0',
        'dstBeginIP': '',
        'dstCity': '',
        'dstCountryCode': '',
        'dstEndIP': '',
        'dstHostname': '',
        'dstISP_AS': '',
        'dstLatitude': '0.000000',
        'dstLongitude': '0.000000',
        'dstMACAddress': '00:00:00:00:00:00',
        'dstOrganization': '',
        'dstPort': '5678',
        'dstPostalCode': '',
        'dstRegion': '',
        'dstRegionName': '',
        'dstTimezone': '',
        'dstVLAN': '0',
        'exporterAddress': '10.1.30.201',
        'firstPacketTime': '20/03/2017 12:18:06',
        'ipProtocol': 'UDP',
        'ipVersion': '6',
        'lastPacketTime': '20/03/2017 12:18:06',
        'natInsideDstAddr': '',
        'natInsideSrcAddr': '',
        'netflowVersion': '9',
        'nextHop': 'ff02::1',
        'srcAddr': 'fe80::200:5eff:fe00:163',
        'srcAreaCode': '0',
        'srcAssignedNetmask': '0',
        'srcBeginIP': '',
        'srcCity': '',
        'srcCountryCode': '',
        'srcEndIP': '',
        'srcHostname': '',
        'srcISP_AS': '',
        'srcLatitude': '0.000000',
        'srcLongitude': '0.000000',
        'srcMACAddress': 'd4:ca:6d:06:0c:12',
        'srcOrganization': '',
        'srcPort': '5678',
        'srcPostalCode': '',
        'srcRegion': '',
        'srcRegionName': '',
        'srcTimezone': '',
        'srcVLAN': '0',
        'tcp_flags': '0',
        'templateID': '257',
        'trafficCategory': 'NETWORK OPERATION',
        'trafficClassification': '[UNRATED] '}},
{'Flow11': {'exporterAddress': 'EOL'}}]
```

def GetOtherFlows(user, password, server, port): This functions returns the last N Other IPv4 or IPv6 flows (ICMP, IGMP, etc.) where N is the value defined in the */opt/fl0wer/etc/fl0wer.conf* configuration file variable *flows_to_keep_in_ram*. Its default is 2000. If the variable *enable_nst* in the configuration value is set to no, no valid data is returned.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Flow0': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
            'FlowDuration': '0',
            'FlowID': '1490012052360527',
            'NBAR': 'AppID: 16777217 (1:1) was not found - NBAR data still '
                    'not available',
            'NPAR': 'ICMP Protocol Code: DESTINATION UNREACHABLE',
            'RULE': '',
            'XINFO': '',
            'connBytes': '172',
            'connPackets': '1',
            'dateReceived': '20/03/2017 12:24:32',
            'dstAddr': '10.100.3.20',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': 'd4:ca:6d:0d:ba:e4',
            'dstOrganization': '',
            'dstPort': '0',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.100.1.1',
            'firstPacketTime': '20/03/2017 12:22:18',
            'ipProtocol': 'ICMP',
            'ipVersion': '4',
            'lastPacketTime': '20/03/2017 12:22:18',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '10',
            'nextHop': '10.100.3.20',
            'srcAddr': '10.100.0.1',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': '00:0d:b9:34:16:18',
            'srcOrganization': '',
            'srcPort': '0',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '0',
            'templateID': '259',
            'trafficCategory': 'NETWORK OPERATION',
            'trafficClassification': '[UNRATED] '}},
< … OMISSIS … >
 {'Flow33': {'FlowDirection': 'INTERNAL_TO_INTERNAL',
             'FlowDuration': '0',
             'FlowID': '1490012119253476',
             'NBAR': 'AppID: 3741384717 (13:479) was not found - NBAR data '
                     'still not available',
            'NPAR': 'ICMP Protocol Code: ECHO REPLY',
            'RULE': '',
            'XINFO': '',
            'connBytes': '56',
            'connPackets': '1',
            'dateReceived': '20/03/2017 12:25:39',
            'dstAddr': '10.100.0.1',
            'dstAreaCode': '0',
            'dstAssignedNetmask': '0',
            'dstBeginIP': '',
            'dstCity': '',
            'dstCountryCode': '',
            'dstEndIP': '',
            'dstHostname': '',
            'dstISP_AS': '',
            'dstLatitude': '0.000000',
            'dstLongitude': '0.000000',
            'dstMACAddress': '00:0d:b9:34:16:18',
            'dstOrganization': '',
            'dstPort': '0',
            'dstPostalCode': '',
            'dstRegion': '',
            'dstRegionName': '',
            'dstTimezone': '',
            'dstVLAN': '0',
            'exporterAddress': '10.100.1.1',
            'firstPacketTime': '20/03/2017 12:22:34',
            'ipProtocol': 'ICMP',
            'ipVersion': '4',
            'lastPacketTime': '20/03/2017 12:22:34',
            'natInsideDstAddr': '',
            'natInsideSrcAddr': '',
            'netflowVersion': '10',
            'nextHop': '10.100.1.20',
            'srcAddr': '10.100.2.20',
            'srcAreaCode': '0',
            'srcAssignedNetmask': '0',
            'srcBeginIP': '',
            'srcCity': '',
            'srcCountryCode': '',
            'srcEndIP': '',
            'srcHostname': '',
            'srcISP_AS': '',
            'srcLatitude': '0.000000',
            'srcLongitude': '0.000000',
            'srcMACAddress': 'd4:ca:6d:06:de:06',
            'srcOrganization': '',
            'srcPort': '0',
            'srcPostalCode': '',
            'srcRegion': '',
            'srcRegionName': '',
            'srcTimezone': '',
            'srcVLAN': '0',
            'tcp_flags': '0',
            'templateID': '259',
            'trafficCategory': 'NETWORK OPERATION',
            'trafficClassification': '[UNRATED] '}},
 {'Flow34': {'exporterAddress': 'EOL'}}]
```

IP Cache Functions

def GetIPCache4(user, password, server, port): This functions returns ALL the IP Cache for IPv4 addresses.

►Be careful when using this, since in an environment with very large number of hosts it could return a LOT of data.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'IPCache4_entry_0': {'Areacode': '0',
                       'City': '',
                       'CountryCode': 'TW',
                       'FQDN': '1-34-23-112.hinet-ip.hinet.net',
                       'ICMPTrafficBytes': '0',
                       'ICMPTrafficPackets': '0',
                       'IGMPTrafficBytes': '0',
                       'IGMPTrafficPackets': '0',
                       'IPv4TrafficBytes': '1018',
                       'IPv4TrafficPackets': '11',
                       'IPv6TrafficBytes': '0',
                       'IPv6TrafficPackets': '0',
                       'ISP_AS': 'AS3462 Data Communication Business Group',
                       'Latitude': '23.500000',
                       'Longitude': '121.000000',
                       'OtherTrafficBytes': '0',
                       'OtherTrafficPackets': '0',
                       'PostalCode': '',
                       'Region': '',
                       'Region_Name': '',
                       'TCPTrafficBytes': '0',
                       'TCPTrafficPackets': '0',
                       'Timezone': '',
                       'UDPTrafficBytes': '1018',
                       'UDPTrafficPackets': '11',
                       'assigned_netmask': '15',
                       'bgp_bytes': '0',
                       'bgp_firstseen': 'Never',
                       'bgp_hits': '0',
                       'bgp_lastseen': 'Never',
                       'bgp_packets': '0',
                       'bytesTrafficAuthentication': '0',
                       'bytesTrafficDataStorage': '0',
                       'bytesTrafficDatabase': '0',
                       'bytesTrafficGaming': '0',
                       'bytesTrafficMAIL': '0',
                       'bytesTrafficUnwanted': '0',
                       'bytesTrafficManagement': '0',
                       'bytesTrafficNetworkOperation': '0',
                       'bytesTrafficNetworkTunnel': '0',
                       'bytesTrafficP2P': '1018',
                       'bytesTrafficPrinting': '0',
                       'bytesTrafficSocial': '0',
                       'bytesTrafficSystemsOperation': '0',
                       'bytesTrafficUnclassified': '0',
                       'bytesTrafficVPN': '0',
                       'bytesTrafficVoiceVideo': '0',
                       'bytesTrafficWEB': '0',
                       'count_relationship_authentication': '0',
                       'count_relationship_data_storage': '0',
                       'count_relationship_database': '0',
                       'count_relationship_gaming': '0',
                       'count_relationship_mail': '0',
                       'count_relationship_unwanted': '0',
                       'count_relationship_management': '0',
                       'count_relationship_network_operation': '0',
                       'count_relationship_network_tunnel': '0',
                       'count_relationship_p2p': '0',
                       'count_relationship_printing': '0',
                       'count_relationship_social': '0',
'count_relationship_systems_operation': '0',
'count_relationship_unclassified': '0',
'count_relationship_voice_video': '0',
'count_relationship_vpn': '0',
'count_relationship_web': '0',
'dataStoreIPLocation': 'External IP',
'dst_bytes': '430',
'dst_hits': '3',
'dst_packets': '5',
'eigrp_bytes': '0',
'eigrp_firstseen': 'Never',
'eigrp_hits': '0',
'eigrp_lastseen': 'Never',
'eigrp_packets': '0',
'firstICMPPacketTime': 'Never',
'firstIGMPPacketTime': 'Never',
'firstOtherPacketTime': 'Never',
'firstPacketTime': '19/06/2017 01:44:31',
'firstTCPPacketTime': 'Never',
'firstUDPPacketTime': '19/06/2017 01:44:31',
'from_netflow_bytes': '0',
'from_netflow_firstseen': 'Never',
'from_netflow_hits': '0',
'from_netflow_lastseen': 'Never',
'from_netflow_packets': '0',
'ftp_bytes': '0',
'ftp_firstseen': 'Never',
'ftp_hits': '0',
'ftp_lastseen': 'Never',
'ftp_packets': '0',
'http_bytes': '0',
'http_firstseen': 'Never',
'http_hits': '0',
'http_info': '',
'http_lastseen': 'Never',
'http_packets': '0',
'https_bytes': '0',
'https_firstseen': 'Never',
'https_hits': '0',
'https_info': '',
'https_lastseen': 'Never',
'https_packets': '0',
'ipv4address': '1.34.23.112',
'iscsi_bytes': '0',
'iscsi_firstseen': 'Never',
'iscsi_hits': '0',
'iscsi_lastseen': 'Never',
'iscsi_packets': '0',
'lastICMPPacketTime': 'Never',
'lastIGMPPacketTime': 'Never',
'lastOtherPacketTime': 'Never',
'lastPacketTime': '19/06/2017 01:44:32',
'lastTCPPacketTime': 'Never',
'lastUDPPacketTime': '19/06/2017 01:44:32',
'nfs_bytes': '0',
'nfs_firstseen': 'Never',
'nfs_hits': '0',
'nfs_lastseen': 'Never',
'nfs_packets': '0',
'organization': 'CHTD, Chunghwa Telecom Co., Ltd.',
'ospf_bytes': '0',
'ospf_firstseen': 'Never',
'ospf_hits': '0',
'ospf_lastseen': 'Never',
'ospf_packets': '0',
'rip_bytes': '0',
'rip_firstseen': 'Never',
'rip_hits': '0',
'rip_lastseen': 'Never',
'rip_packets': '0',
'routerIPLocation': 'External IP',
'smb_bytes': '0',
'smb_firstseen': 'Never',
'smb_hits': '0',
'smb_lastseen': 'Never',
'smb_packets': '0',
'snmpagent_bytes': '0',
'snmpagent_firstseen': 'Never',
                       'snmpagent_hits': '0',
                       'snmpagent_lastseen': 'Never',
                       'snmpagent_packets': '0',
                       'snmpclient_bytes': '0',
                       'snmpclient_firstseen': 'Never',
                       'snmpclient_hits': '0',
                       'snmpclient_lastseen': 'Never',
                       'snmpclient_packets': '0',
                       'social_bytes': '0',
                       'social_firstseen': 'Never',
                       'social_hits': '0',
                       'social_info': '',
                       'social_lastseen': 'Never',
                       'social_packets': '0',
                       'src_bytes': '588',
                       'src_hits': '4',
                       'src_packets': '6',
                       'subnet_beginip': '1.34.0.0',
                       'subnet_endip': '1.35.255.255',
                       'tcp_ACK_count': '0',
                       'tcp_CWR_count': '0',
                       'tcp_ECN_count': '0',
                       'tcp_FIN_count': '0',
                       'tcp_PSH_count': '0',
                       'tcp_RST_count': '0',
                       'tcp_SYN_count': '0',
                       'tcp_URG_count': '0',
                       'tftp_bytes': '0',
                       'tftp_firstseen': 'Never',
                       'tftp_hits': '0',
                       'tftp_lastseen': 'Never',
                       'tftp_packets': '0',
                       'trafficAuthentication': '0',
                       'trafficDataStorage': '0',
                       'trafficDatabase': '0',
                       'trafficGaming': '0',
                       'trafficMAIL': '0',
                       'trafficUnwanted': '0',
                       'trafficManagement': '0',
                       'trafficNetworkOperation': '0',
                       'trafficNetworkTunnel': '0',
                       'trafficP2P': '11',
                       'trafficPrinting': '0',
                       'trafficSocial': '0',
                       'trafficSystemsOperation': '0',
                       'trafficUnclassified': '0',
                       'trafficVPN': '0',
                       'trafficVoiceVideo': '0',
                       'trafficWEB': '0',
                       'webLocation': 'External IP'}},
< … OMISSIS … >
 {'IPCache4_entry_157': {'Areacode': '0',
                         'City': '',
                         'CountryCode': '',
                         'FQDN': '239.255.255.250',
                         'ICMPTrafficBytes': '0',
                         'ICMPTrafficPackets': '0',
                         'IGMPTrafficBytes': '0',
                         'IGMPTrafficPackets': '0',
                         'IPv4TrafficBytes': '1000',
                         'IPv4TrafficPackets': '5',
                         'IPv6TrafficBytes': '0',
                         'IPv6TrafficPackets': '0',
                         'ISP_AS': '',
                         'Latitude': '0.000000',
                         'Longitude': '0.000000',
                         'OtherTrafficBytes': '0',
                         'OtherTrafficPackets': '0',
                         'PostalCode': '',
                         'Region': '',
                         'Region_Name': '',
                         'TCPTrafficBytes': '0',
                         'TCPTrafficPackets': '0',
                         'Timezone': '',
                         'UDPTrafficBytes': '1000',
                         'UDPTrafficPackets': '5',
                         'assigned_netmask': '0',
                         'bgp_bytes': '0',
'bgp_firstseen': 'Never',
'bgp_hits': '0',
'bgp_lastseen': 'Never',
'bgp_packets': '0',
'bytesTrafficAuthentication': '0',
'bytesTrafficDataStorage': '0',
'bytesTrafficDatabase': '0',
'bytesTrafficGaming': '0',
'bytesTrafficMAIL': '0',
'bytesTrafficUnwanted': '0',
'bytesTrafficManagement': '0',
'bytesTrafficNetworkOperation': '1000',
'bytesTrafficNetworkTunnel': '0',
'bytesTrafficP2P': '0',
'bytesTrafficPrinting': '0',
'bytesTrafficSocial': '0',
'bytesTrafficSystemsOperation': '0',
'bytesTrafficUnclassified': '0',
'bytesTrafficVPN': '0',
'bytesTrafficVoiceVideo': '0',
'bytesTrafficWEB': '0',
'count_relationship_authentication': '0',
'count_relationship_data_storage': '0',
'count_relationship_database': '0',
'count_relationship_gaming': '0',
'count_relationship_mail': '0',
'count_relationship_unwanted': '0',
'count_relationship_management': '0',
'count_relationship_network_operation': '0',
'count_relationship_network_tunnel': '0',
'count_relationship_p2p': '0',
'count_relationship_printing': '0',
'count_relationship_social': '0',
'count_relationship_systems_operation': '0',
'count_relationship_unclassified': '0',
'count_relationship_voice_video': '0',
'count_relationship_vpn': '0',
'count_relationship_web': '0',
'dataStoreIPLocation': 'External IP',
'dst_bytes': '1000',
'dst_hits': '1',
'dst_packets': '5',
'eigrp_bytes': '0',
'eigrp_firstseen': 'Never',
'eigrp_hits': '0',
'eigrp_lastseen': 'Never',
'eigrp_packets': '0',
'firstICMPPacketTime': 'Never',
'firstIGMPPacketTime': 'Never',
'firstOtherPacketTime': 'Never',
'firstPacketTime': '19/06/2017 01:46:19',
'firstTCPPacketTime': 'Never',
'firstUDPPacketTime': '19/06/2017 01:46:19',
'from_netflow_bytes': '0',
'from_netflow_firstseen': 'Never',
'from_netflow_hits': '0',
'from_netflow_lastseen': 'Never',
'from_netflow_packets': '0',
'ftp_bytes': '0',
'ftp_firstseen': 'Never',
'ftp_hits': '0',
'ftp_lastseen': 'Never',
'ftp_packets': '0',
'http_bytes': '0',
'http_firstseen': 'Never',
'http_hits': '0',
'http_info': '',
'http_lastseen': 'Never',
'http_packets': '0',
'https_bytes': '0',
'https_firstseen': 'Never',
'https_hits': '0',
'https_info': '',
'https_lastseen': 'Never',
'https_packets': '0',
'ipv4address': '239.255.255.250',
'iscsi_bytes': '0',
'iscsi_firstseen': 'Never',
'iscsi_hits': '0',
'iscsi_lastseen': 'Never',
'iscsi_packets': '0',
'lastICMPPacketTime': 'Never',
'lastIGMPPacketTime': 'Never',
'lastOtherPacketTime': 'Never',
'lastPacketTime': '19/06/2017 01:46:19',
'lastTCPPacketTime': 'Never',
'lastUDPPacketTime': '19/06/2017 01:46:19',
'nfs_bytes': '0',
'nfs_firstseen': 'Never',
'nfs_hits': '0',
'nfs_lastseen': 'Never',
'nfs_packets': '0',
'organization': '',
'ospf_bytes': '0',
'ospf_firstseen': 'Never',
'ospf_hits': '0',
'ospf_lastseen': 'Never',
'ospf_packets': '0',
'rip_bytes': '0',
'rip_firstseen': 'Never',
'rip_hits': '0',
'rip_lastseen': 'Never',
'rip_packets': '0',
'routerIPLocation': 'External IP',
'smb_bytes': '0',
'smb_firstseen': 'Never',
'smb_hits': '0',
'smb_lastseen': 'Never',
'smb_packets': '0',
'snmpagent_bytes': '0',
'snmpagent_firstseen': 'Never',
'snmpagent_hits': '0',
'snmpagent_lastseen': 'Never',
'snmpagent_packets': '0',
'snmpclient_bytes': '0',
'snmpclient_firstseen': 'Never',
'snmpclient_hits': '0',
'snmpclient_lastseen': 'Never',
'snmpclient_packets': '0',
'social_bytes': '0',
'social_firstseen': 'Never',
'social_hits': '0',
'social_info': '',
'social_lastseen': 'Never',
'social_packets': '0',
'src_bytes': '0',
'src_hits': '0',
'src_packets': '0',
'subnet_beginip': '',
'subnet_endip': '',
'tcp_ACK_count': '0',
'tcp_CWR_count': '0',
'tcp_ECN_count': '0',
'tcp_FIN_count': '0',
'tcp_PSH_count': '0',
'tcp_RST_count': '0',
'tcp_SYN_count': '0',
'tcp_URG_count': '0',
'tftp_bytes': '0',
'tftp_firstseen': 'Never',
'tftp_hits': '0',
'tftp_lastseen': 'Never',
'tftp_packets': '0',
'trafficAuthentication': '0',
'trafficDataStorage': '0',
'trafficDatabase': '0',
'trafficGaming': '0',
'trafficMAIL': '0',
'trafficManagement': '0',
'trafficNetworkOperation': '5',
'trafficNetworkTunnel': '0',
'trafficP2P': '0',
'trafficPrinting': '0',
'trafficSocial': '0',
'trafficSystemsOperation': '0',
'trafficUnclassified': '0',
                          'trafficUnwanted': '0',
                          'trafficVPN': '0',
                          'trafficVoiceVideo': '0',
                          'trafficWEB': '0',
                          'webLocation': 'External IP'}},
 {'IPCache4_entry_EOF': {'IPCache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopSrcHits(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv4 sorted by absolute flow count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'dnsmaster.ip6net.me',
                       'dst_bytes': '112968731',
                       'dst_hits': '803873',
                       'dst_packets': '1608843',
                       'ipv4address': '10.1.30.150',
                       'src_bytes': '179295068',
                       'src_hits': '1085978',
                       'src_packets': '1512829'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': '195-154-179-2.rev.poneytelecom.eu',
                        'dst_bytes': '224326',
                        'dst_hits': '1515',
                        'dst_packets': '2333',
                        'ipv4address': '195.154.179.2',
                        'src_bytes': '190757',
                        'src_hits': '1349',
                        'src_packets': '1916'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopSrcPackets(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv4 sorted by absolute packet count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'firewall.ip6net.me',
                       'dst_bytes': '630054844',
                       'dst_hits': '233',
                       'dst_packets': '1207228',
                       'ipv4address': '10.1.30.100',
                       'src_bytes': '1720740694',
                       'src_hits': '15035',
                       'src_packets': '2333305'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': 'ntp2.inrim.it',
                        'dst_bytes': '546562',
                        'dst_hits': '4867',
                        'dst_packets': '7180',
                        'ipv4address': '193.204.114.233',
                        'src_bytes': '466368',
                        'src_hits': '3919',
                        'src_packets': '6125'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopSrcBytes(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv4 sorted by absolute byte count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'firewall.ip6net.me',
                       'dst_bytes': '630054844',
                       'dst_hits': '233',
                       'dst_packets': '1207228',
                       'ipv4address': '10.1.30.100',
                       'src_bytes': '17212300740694',
                       'src_hits': '15035',
                       'src_packets': '2333305'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': 'ntp2.inrim.it',
                        'dst_bytes': '546562',
                        'dst_hits': '4867',
                        'dst_packets': '7180',
                        'ipv4address': '193.204.114.233',
                        'src_bytes': '466368',
                        'src_hits': '3919',
                        'src_packets': '6125'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopDstHits(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv4 sorted by absolute flow count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'firewall.ip6net.me',
                       'dst_bytes': '63001233354844',
                       'dst_hits': '233',
                       'dst_packets': '1207228',
                       'ipv4address': '10.1.30.100',
                       'src_bytes': '172694',
                       'src_hits': '15035',
                       'src_packets': '2333305'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': 'ntp2.inrim.it',
                        'dst_bytes': '546562',
                        'dst_hits': '4867',
                        'dst_packets': '7180',
                        'ipv4address': '193.204.114.233',
                        'src_bytes': '466368',
                        'src_hits': '3919',
                        'src_packets': '6125'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopDstPackets(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv4 sorted by absolute packet count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'firewall.ip6net.me',
                       'dst_bytes': '63001233354844',
                       'dst_hits': '233',
                       'dst_packets': '120235656228',
                       'ipv4address': '10.1.30.100',
                       'src_bytes': '172694',
                       'src_hits': '15035',
                       'src_packets': '2333305'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': 'ntp2.inrim.it',
                        'dst_bytes': '546562',
                        'dst_hits': '4867',
                        'dst_packets': '7180',
                        'ipv4address': '193.204.114.233',
                        'src_bytes': '466368',
                        'src_hits': '3919',
                        'src_packets': '6125'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache4TopDstBytes(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv4 sorted by absolute byte count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache4_entry_0': {'FQDN': 'firewall.ip6net.me',
                       'dst_bytes': '63001233354844',
                       'dst_hits': '233',
                       'dst_packets': '12056228',
                       'ipv4address': '10.1.30.100',
                       'src_bytes': '172694',
                       'src_hits': '15035',
                       'src_packets': '2333305'}},
< … OMISSIS … >
 {'ipcache4_entry_99': {'FQDN': 'ntp2.inrim.it',
                        'dst_bytes': '546562',
                        'dst_hits': '4867',
                        'dst_packets': '7180',
                        'ipv4address': '193.204.114.233',
                        'src_bytes': '466368',
                        'src_hits': '3919',
                        'src_packets': '6125'}},
 {'ipcache4_entry_EOF': {'ipcache4_entry_EOF': 'EOF'}}]
```

def GetIPCache6(user, password, server, port): This functions returns ALL the IP Cache for IPv6 addresses. Be careful when using this, since in an environment with very large number of hosts it could return a LOT of data.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'IPCache6_entry_0': {'Areacode': '0',
                       'City': '',
                       'CountryCode': '',
                       'FQDN': '2001:0:9d38:6ab8:2c:36af:da6c:d8e1',
                       'ICMPTrafficBytes': '0',
                       'ICMPTrafficPackets': '0',
                       'IGMPTrafficBytes': '0',
                       'IGMPTrafficPackets': '0',
                       'IPv4TrafficBytes': '0',
                       'IPv4TrafficPackets': '0',
                       'IPv6TrafficBytes': '0',
                       'IPv6TrafficPackets': '0',
                       'ISP_AS': '',
                       'Latitude': '0.000000',
                       'Longitude': '0.000000',
                       'OtherTrafficBytes': '0',
                       'OtherTrafficPackets': '0',
                       'PostalCode': '',
                       'Region': '',
                       'Region_Name': '',
                       'TCPTrafficBytes': '0',
                       'TCPTrafficPackets': '0',
                       'Timezone': '',
                       'UDPTrafficBytes': '0',
                       'UDPTrafficPackets': '0',
                       'assigned_netmask': '0',
                       'bgp_bytes': '0',
                       'bgp_firstseen': 'Never',
                       'bgp_hits': '0',
                       'bgp_lastseen': 'Never',
                       'bgp_packets': '0',
                       'bytesTrafficAuthentication': '0',
                       'bytesTrafficDataStorage': '0',
                       'bytesTrafficDatabase': '0',
                       'bytesTrafficGaming': '0',
                       'bytesTrafficMAIL': '0',
                       'bytesTrafficUnwanted': '0',
                       'bytesTrafficManagement': '0',
                       'bytesTrafficNetworkOperation': '0',
                       'bytesTrafficNetworkTunnel': '0',
                       'bytesTrafficP2P': '0',
                       'bytesTrafficPrinting': '0',
                       'bytesTrafficSocial': '0',
                       'bytesTrafficSystemsOperation': '0',
                       'bytesTrafficUnclassified': '0',
                       'bytesTrafficVPN': '0',
                       'bytesTrafficVoiceVideo': '0',
                       'bytesTrafficWEB': '0',
                       'count_relationship_authentication': '0',
                       'count_relationship_data_storage': '0',
                       'count_relationship_database': '0',
                       'count_relationship_gaming': '0',
                       'count_relationship_mail': '0',
                       'count_relationship_unwanted': '0',
                       'count_relationship_management': '0',
                       'count_relationship_network_operation': '0',
                       'count_relationship_network_tunnel': '0',
                       'count_relationship_p2p': '0',
                       'count_relationship_printing': '0',
                       'count_relationship_social': '0',
                       'count_relationship_systems_operation': '0',
                       'count_relationship_unclassified': '0',
                       'count_relationship_voice_video': '0',
                       'count_relationship_vpn': '0',
                       'count_relationship_web': '0',
                       'dataStoreIPLocation': 'External IP',
                       'dst_bytes': '106',
                       'dst_hits': '1',
                       'dst_packets': '1',
                       'eigrp_bytes': '0',
                       'eigrp_firstseen': 'Never',
                       'eigrp_hits': '0',
                       'eigrp_lastseen': 'Never',
                       'eigrp_packets': '0',
                       'firstICMPPacketTime': 'Never',
                       'firstIGMPPacketTime': 'Never',
                       'firstOtherPacketTime': 'Never',
                       'firstPacketTime': 'Never',
                       'firstTCPPacketTime': 'Never',
                       'firstUDPPacketTime': 'Never',
                       'from_netflow_bytes': '0',
                       'from_netflow_firstseen': 'Never',
                       'from_netflow_hits': '0',
                       'from_netflow_lastseen': 'Never',
                       'from_netflow_packets': '0',
                       'ftp_bytes': '0',
                       'ftp_firstseen': 'Never',
                       'ftp_hits': '0',
                       'ftp_lastseen': 'Never',
                       'ftp_packets': '0',
                       'http_bytes': '0',
                       'http_firstseen': 'Never',
                       'http_hits': '0',
                       'http_info': '',
                       'http_lastseen': 'Never',
                       'http_packets': '0',
                       'https_bytes': '0',
                       'https_firstseen': 'Never',
                       'https_hits': '0',
                       'https_info': '',
                       'https_lastseen': 'Never',
                       'https_packets': '0',
                       'ipv6address': '2001:0:9d38:6ab8:2c:36af:da6c:d8e1',
                       'iscsi_bytes': '0',
                       'iscsi_firstseen': 'Never',
                       'iscsi_hits': '0',
                       'iscsi_lastseen': 'Never',
                       'iscsi_packets': '0',
                       'lastICMPPacketTime': 'Never',
                       'lastIGMPPacketTime': 'Never',
                       'lastOtherPacketTime': 'Never',
                       'lastPacketTime': 'Never',
                       'lastTCPPacketTime': 'Never',
                       'lastUDPPacketTime': 'Never',
                       'nfs_bytes': '0',
                       'nfs_firstseen': 'Never',
                       'nfs_hits': '0',
                       'nfs_lastseen': 'Never',
                       'nfs_packets': '0',
                       'organization': '',
                       'ospf_bytes': '0',
                       'ospf_firstseen': 'Never',
                       'ospf_hits': '0',
                       'ospf_lastseen': 'Never',
                       'ospf_packets': '0',
                       'rip_bytes': '0',
                       'rip_firstseen': 'Never',
                       'rip_hits': '0',
                       'rip_lastseen': 'Never',
                       'rip_packets': '0',
                       'routerIPLocation': 'External IP',
                       'smb_bytes': '0',
                       'smb_firstseen': 'Never',
                       'smb_hits': '0',
                       'smb_lastseen': 'Never',
                       'smb_packets': '0',
                       'snmpagent_bytes': '0',
                       'snmpagent_firstseen': 'Never',
                       'snmpagent_hits': '0',
                       'snmpagent_lastseen': 'Never',
                       'snmpagent_packets': '0',
                       'snmpclient_bytes': '0',
                       'snmpclient_firstseen': 'Never',
                       'snmpclient_hits': '0',
                       'snmpclient_lastseen': 'Never',
                       'snmpclient_packets': '0',
                       'social_bytes': '0',
                                              'social_firstseen': 'Never',
                                              'social_hits': '0',
                                              'social_info': '',
                                              'social_lastseen': 'Never',
                                              'social_packets': '0',
                                              'src_bytes': '0',
                                              'src_hits': '0',
                                              'src_packets': '0',
                                              'subnet_beginip': '',
                                              'subnet_endip': '',
                                              'tcp_ACK_count': '0',
                                              'tcp_CWR_count': '0',
                                              'tcp_ECN_count': '0',
                                              'tcp_FIN_count': '0',
                                              'tcp_PSH_count': '0',
                                              'tcp_RST_count': '0',
                                              'tcp_SYN_count': '0',
                                              'tcp_URG_count': '0',
                                              'tftp_bytes': '0',
                                              'tftp_firstseen': 'Never',
                                              'tftp_hits': '0',
                                              'tftp_lastseen': 'Never',
                                              'tftp_packets': '0',
                                              'trafficAuthentication': '0',
                                              'trafficDataStorage': '0',
                                              'trafficDatabase': '0',
                                              'trafficGaming': '0',
                                              'trafficMAIL': '0',
                                              'trafficUnwanted': '0',
                                              'trafficManagement': '0',
                                              'trafficNetworkOperation': '0',
                                              'trafficNetworkTunnel': '0',
                                              'trafficP2P': '0',
                                              'trafficPrinting': '0',
                                              'trafficSocial': '0',
                                              'trafficSystemsOperation': '0',
                                              'trafficUnclassified': '0',
                                              'trafficVPN': '0',
                                              'trafficVoiceVideo': '0',
                                              'trafficWEB': '0',
                                              'webLocation': 'External IP'}},
                       < … OMISSIS … >
                        {'IPCache6_entry_11': {'Areacode': '0',
                                               'City': '',
                                               'CountryCode': '',
                                               'FQDN': '2a02:a449:72cc:1:211:32ff:fe27:9030',
                                               'ICMPTrafficBytes': '0',
                                               'ICMPTrafficPackets': '0',
                                               'IGMPTrafficBytes': '0',
                                               'IGMPTrafficPackets': '0',
                                               'IPv4TrafficBytes': '0',
                                               'IPv4TrafficPackets': '0',
                                               'IPv6TrafficBytes': '0',
                                               'IPv6TrafficPackets': '0',
                                               'ISP_AS': '',
                                               'Latitude': '0.000000',
                                               'Longitude': '0.000000',
                                               'OtherTrafficBytes': '0',
                                               'OtherTrafficPackets': '0',
                                               'PostalCode': '',
                                               'Region': '',
                                               'Region_Name': '',
                                               'TCPTrafficBytes': '0',
                                               'TCPTrafficPackets': '0',
                                               'Timezone': '',
                                               'UDPTrafficBytes': '0',
                                               'UDPTrafficPackets': '0',
                                               'assigned_netmask': '0',
                                               'bgp_bytes': '0',
                                               'bgp_firstseen': 'Never',
                                               'bgp_hits': '0',
                                               'bgp_lastseen': 'Never',
                                               'bgp_packets': '0',
                                               'bytesTrafficAuthentication': '0',
                                               'bytesTrafficDataStorage': '0',
                                               'bytesTrafficDatabase': '0',
                                               'bytesTrafficGaming': '0',
                                               'bytesTrafficMAIL': '0',
                       'bytesTrafficUnwanted': '0',
                       'bytesTrafficManagement': '0',
                       'bytesTrafficNetworkOperation': '0',
                       'bytesTrafficNetworkTunnel': '0',
                       'bytesTrafficP2P': '0',
                       'bytesTrafficPrinting': '0',
                       'bytesTrafficSocial': '0',
                       'bytesTrafficSystemsOperation': '0',
                       'bytesTrafficUnclassified': '0',
                       'bytesTrafficVPN': '0',
                       'bytesTrafficVoiceVideo': '0',
                       'bytesTrafficWEB': '0',
                       'count_relationship_authentication': '0',
                       'count_relationship_data_storage': '0',
                       'count_relationship_database': '0',
                       'count_relationship_gaming': '0',
                       'count_relationship_mail': '0',
                       'count_relationship_unwanted': '0',
                       'count_relationship_management': '0',
                       'count_relationship_network_operation': '0',
                       'count_relationship_network_tunnel': '0',
                       'count_relationship_p2p': '0',
                       'count_relationship_printing': '0',
                       'count_relationship_social': '0',
                       'count_relationship_systems_operation': '0',
                       'count_relationship_unclassified': '0',
                       'count_relationship_voice_video': '0',
                       'count_relationship_vpn': '0',
                       'count_relationship_web': '0',
                       'dataStoreIPLocation': 'External IP',
                       'dst_bytes': '106',
                       'dst_hits': '1',
                       'dst_packets': '1',
                       'eigrp_bytes': '0',
                       'eigrp_firstseen': 'Never',
                       'eigrp_hits': '0',
                       'eigrp_lastseen': 'Never',
                       'eigrp_packets': '0',
                       'firstICMPPacketTime': 'Never',
                       'firstIGMPPacketTime': 'Never',
                       'firstOtherPacketTime': 'Never',
                       'firstPacketTime': 'Never',
                       'firstTCPPacketTime': 'Never',
                       'firstUDPPacketTime': 'Never',
                       'from_netflow_bytes': '0',
                       'from_netflow_firstseen': 'Never',
                       'from_netflow_hits': '0',
                       'from_netflow_lastseen': 'Never',
                       'from_netflow_packets': '0',
                       'ftp_bytes': '0',
                       'ftp_firstseen': 'Never',
                       'ftp_hits': '0',
                       'ftp_lastseen': 'Never',
                       'ftp_packets': '0',
                       'http_bytes': '0',
                       'http_firstseen': 'Never',
                       'http_hits': '0',
                       'http_info': '',
                       'http_lastseen': 'Never',
                       'http_packets': '0',
                       'https_bytes': '0',
                       'https_firstseen': 'Never',
                       'https_hits': '0',
                       'https_info': '',
                       'https_lastseen': 'Never',
                       'https_packets': '0',
                       'ipv6address': '2a02:a449:72cc:1:211:32ff:fe27:9030',
                       'iscsi_bytes': '0',
                       'iscsi_firstseen': 'Never',
                       'iscsi_hits': '0',
                       'iscsi_lastseen': 'Never',
                       'iscsi_packets': '0',
                       'lastICMPPacketTime': 'Never',
                       'lastIGMPPacketTime': 'Never',
                       'lastOtherPacketTime': 'Never',
                       'lastPacketTime': 'Never',
                       'lastTCPPacketTime': 'Never',
                       'lastUDPPacketTime': 'Never',
                                              'nfs_bytes': '0',
                                              'nfs_firstseen': 'Never',
                                              'nfs_hits': '0',
                                              'nfs_lastseen': 'Never',
                                              'nfs_packets': '0',
                                              'organization': '',
                                              'ospf_bytes': '0',
                                              'ospf_firstseen': 'Never',
                                              'ospf_hits': '0',
                                              'ospf_lastseen': 'Never',
                                              'ospf_packets': '0',
                                              'rip_bytes': '0',
                                              'rip_firstseen': 'Never',
                                              'rip_hits': '0',
                                              'rip_lastseen': 'Never',
                                              'rip_packets': '0',
                                              'routerIPLocation': 'External IP',
                                              'smb_bytes': '0',
                                              'smb_firstseen': 'Never',
                                              'smb_hits': '0',
                                              'smb_lastseen': 'Never',
                                              'smb_packets': '0',
                                              'snmpagent_bytes': '0',
                                              'snmpagent_firstseen': 'Never',
                                              'snmpagent_hits': '0',
                                              'snmpagent_lastseen': 'Never',
                                              'snmpagent_packets': '0',
                                              'snmpclient_bytes': '0',
                                              'snmpclient_firstseen': 'Never',
                                              'snmpclient_hits': '0',
                                              'snmpclient_lastseen': 'Never',
                                              'snmpclient_packets': '0',
                                              'social_bytes': '0',
                                              'social_firstseen': 'Never',
                                              'social_hits': '0',
                                              'social_info': '',
                                              'social_lastseen': 'Never',
                                              'social_packets': '0',
                                              'src_bytes': '0',
                                              'src_hits': '0',
                                              'src_packets': '0',
                                              'subnet_beginip': '',
                                              'subnet_endip': '',
                                              'tcp_ACK_count': '0',
                                              'tcp_CWR_count': '0',
                                              'tcp_ECN_count': '0',
                                              'tcp_FIN_count': '0',
                                              'tcp_PSH_count': '0',
                                              'tcp_RST_count': '0',
                                              'tcp_SYN_count': '0',
                                              'tcp_URG_count': '0',
                                              'tftp_bytes': '0',
                                              'tftp_firstseen': 'Never',
                                              'tftp_hits': '0',
                                              'tftp_lastseen': 'Never',
                                              'tftp_packets': '0',
                                              'trafficAuthentication': '0',
                                              'trafficDataStorage': '0',
                                              'trafficDatabase': '0',
                                              'trafficGaming': '0',
                                              'trafficMAIL': '0',
                                              'trafficUnwanted': '0',
                                              'trafficManagement': '0',
                                              'trafficNetworkOperation': '0',
                                              'trafficNetworkTunnel': '0',
                                              'trafficP2P': '0',
                                              'trafficPrinting': '0',
                                              'trafficSocial': '0',
                                              'trafficSystemsOperation': '0',
                                              'trafficUnclassified': '0',
                                              'trafficVPN': '0',
                                              'trafficVoiceVideo': '0',
                                              'trafficWEB': '0',
                                              'webLocation': 'External IP'}},
                        {'IPCache6_entry_EOF': {'IPCache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopSrcHits(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv6 sorted by absolute flow count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '1485838',
                       'dst_hits': '8353',
                       'dst_packets': '14908',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '1262529',
                       'src_hits': '9895',
                       'src_packets': '10770'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopSrcPackets(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv6 sorted by absolute packet count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '1485838',
                       'dst_hits': '8353',
                       'dst_packets': '14908',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '1262529',
                       'src_hits': '9895',
                       'src_packets': '107123332170'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopSrcBytes(user, password, server, port): This function returns the top 100 source IP Cache entries for IPv6 sorted by absolute byte count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '1485838',
                       'dst_hits': '8353',
                       'dst_packets': '14908',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '12629893475529',
                       'src_hits': '9895',
                       'src_packets': '107120'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopDstHits(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv6 sorted by absolute flow count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '1485838',
                       'dst_hits': '83237843453',
                       'dst_packets': '1491508',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '1262929',
                       'src_hits': '9895',
                       'src_packets': '107120'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopDstPackets(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv6 sorted by absolute packet count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '1485838',
                       'dst_hits': '8353',
                       'dst_packets': '14912213123308',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '1262929',
                       'src_hits': '9895',
                       'src_packets': '107120'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPCache6TopDstBytes(user, password, server, port): This function returns the top 100 destination IP Cache entries for IPv6 sorted by absolute byte count.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'ipcache6_entry_0': {'FQDN': '2001:470:8b5b:0:10:1:30:210',
                       'dst_bytes': '14858382131233',
                       'dst_hits': '8353',
                       'dst_packets': '1491208',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:210',
                       'src_bytes': '1262929',
                       'src_hits': '9895',
                       'src_packets': '107120'}},
< … OMISSIS … >
 {'ipcache6_entry_99': {'FQDN': '2001:16d8:dd00:8225::2',
                        'dst_bytes': '6468',
                        'dst_hits': '60',
                        'dst_packets': '60',
                        'ipv6address': '2001:16d8:dd00:8225::2',
                        'src_bytes': '3154',
                        'src_hits': '26',
                        'src_packets': '26'}},
 {'ipcache6_entry_EOF': {'ipcache6_entry_EOF': 'EOF'}}]
```

def GetIPInfo(user, password, server, port,ip): Gets the details of the single specified IP address (the IP can be an IPv4 or IPv6 address).

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'IPCache_single_entry': {'Areacode': '0',
                           'City': '',
                           'CountryCode': '',
                           'FQDN': 'fw1.ip6net.me',
                           'ICMPTrafficBytes': '0',
                           'ICMPTrafficPackets': '0',
                           'IGMPTrafficBytes': '0',
                           'IGMPTrafficPackets': '0',
                           'IPv4TrafficBytes': '3994012',
                           'IPv4TrafficPackets': '52731',
                           'IPv6TrafficBytes': '0',
                           'IPv6TrafficPackets': '0',
                           'ISP_AS': '',
                           'Latitude': '0.000000',
                           'Longitude': '0.000000',
                           'OtherTrafficBytes': '0',
                           'OtherTrafficPackets': '0',
                           'PostalCode': '',
                           'Region': '',
                           'Region_Name': '',
                           'TCPTrafficBytes': '175801',
                           'TCPTrafficPackets': '2404',
                           'Timezone': '',
                           'UDPTrafficBytes': '3818211',
                           'UDPTrafficPackets': '50327',
                           'assigned_netmask': '0',
                           'bytesTrafficAuthentication': '0',
                           'bytesTrafficDataStorage': '282',
                           'bytesTrafficDatabase': '0',
                           'bytesTrafficGaming': '702',
                           'bytesTrafficMAIL': '0',
                           'bytesTrafficManagement': '2041731',
                           'bytesTrafficNetworkOperation': '1934930',
                           'bytesTrafficNetworkTunnel': '0',
                           'bytesTrafficP2P': '74',
                           'bytesTrafficPrinting': '0',
                           'bytesTrafficSocial': '0',
                           'bytesTrafficSystemsOperation': '0',
                           'bytesTrafficUnclassified': '0',
                           'bytesTrafficVPN': '0',
                           'bytesTrafficVoiceVideo': '2530',
                           'bytesTrafficWEB': '13763',
                           'dst_bytes': '1397069',
                           'dst_hits': '9818',
                           'dst_packets': '18829',
                           'firstICMPPacketTime': '01/01/1970 01:00:00',
                           'firstIGMPPacketTime': '01/01/1970 01:00:00',
                           'firstOtherPacketTime': '01/01/1970 01:00:00',
                           'firstPacketTime': '20/03/2017 00:01:38',
                           'firstTCPPacketTime': '20/03/2017 00:27:23',
                           'firstUDPPacketTime': '20/03/2017 00:01:38',
                           'ipv4Address': '10.1.30.101',
                           'lastICMPPacketTime': '01/01/1970 01:00:00',
                           'lastIGMPPacketTime': '01/01/1970 01:00:00',
                           'lastOtherPacketTime': '01/01/1970 01:00:00',
                           'lastPacketTime': '20/03/2017 18:37:03',
                           'lastTCPPacketTime': '20/03/2017 18:37:03',
                           'lastUDPPacketTime': '20/03/2017 18:36:48',
                           'organization': '',
                           'src_bytes': '2596943',
                           'src_hits': '19464',
                           'src_packets': '33902',
                           'subnet_beginip': '',
                           'subnet_endip': '',
                           'tcp_ACK_count': '762',
                           'tcp_CWR_count': '0',
                           'tcp_ECN_count': '1',
                           'tcp_FIN_count': '24',
                           'tcp_PSH_count': '362',
                           'tcp_RST_count': '9',
                           'tcp_SYN_count': '61',
                                'tcp_URG_count': '1',
                                'trafficAuthentication': '0',
                                'trafficDataStorage': '3',
                                'trafficDatabase': '0',
                                'trafficGaming': '9',
                                'trafficMAIL': '0',
                                'trafficManagement': '27161',
                                'trafficNetworkOperation': '25264',
                                'trafficNetworkTunnel': '0',
                                'trafficP2P': '1',
                                'trafficPrinting': '0',
                                'trafficSocial': '0',
                                'trafficSystemsOperation': '0',
                                'trafficUnclassified': '0',
                                'trafficVPN': '0',
                                'trafficVoiceVideo': '34',
                                'trafficWEB': '259'}}]
```

def GetRelationInfo(user, password, server, port,ip): Gets the relations of the specified IP with other IP addresses of the same family (the IP can be an IPv4 or IPv6 address)

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'Relations_single_entry': {'FQDN': 'graphite.ip6net.me',
                             'dateExport': '20/03/2017 18:38:14',
                             'ipAddress': '10.1.30.210',
                             'numRelations': '4'}},
 {'Relationship_0': {'City': '',
                     'CountryCode': '',
                     'ICMPTrafficBytes': '0',
                     'ICMPTrafficPackets': '0',
                     'IPv4TrafficBytes': '48746900',
                     'IPv4TrafficPackets': '97839',
                     'IPv6TrafficBytes': '0',
                     'IPv6TrafficPackets': '97832',
                     'Organization': 'Management',
                     'OtherTrafficBytes': '0',
                     'OtherTrafficPackets': '0',
                     'RegionCode': '',
                     'RegionName': '',
                     'TCPTrafficBytes': '48746900',
                     'TCPTrafficPackets': '97839',
                     'TargetAddress': '10.1.30.220',
                     'TargetFQDN': 'data.ip6net.me',
                     'UDPTrafficBytes': '0',
                     'UDPTrafficPackets': '0',
                     'bytesTrafficAuthentication': '8712',
                     'bytesTrafficDataStorage': '13053',
                     'bytesTrafficDatabase': '12614',
                     'bytesTrafficGaming': '80',
                     'bytesTrafficMAIL': '0',
                     'bytesTrafficManagement': '48633509',
                     'bytesTrafficNetworkOperation': '0',
                     'bytesTrafficNetworkTunnel': '4064',
                     'bytesTrafficP2P': '0',
                     'bytesTrafficPrinting': '0',
                     'bytesTrafficSocial': '0',
                     'bytesTrafficSystemsOperation': '0',
                     'bytesTrafficUnclassified': '0',
                     'bytesTrafficVPN': '0',
                     'bytesTrafficVoiceVideo': '36470',
                     'bytesTrafficWEB': '0',
                     'packetsTrafficAuthentication': '16',
                     'packetsTrafficDataStorage': '24',
                     'packetsTrafficDatabase': '25',
                     'packetsTrafficGaming': '38478',
                     'packetsTrafficMAIL': '0',
                     'packetsTrafficManagement': '97601',
                     'packetsTrafficNetworkOperation': '0',
                     'packetsTrafficNetworkTunnel': '8',
                     'packetsTrafficP2P': '0',
                     'packetsTrafficPrinting': '0',
                     'packetsTrafficSocial': '0',
                     'packetsTrafficSystemsOperation': '0',
                     'packetsTrafficUnclassified': '0',
                     'packetsTrafficVPN': '0',
                     'packetsTrafficVoiceVideo': '85',
                     'packetsTrafficWEB': '0',
                     'tcp_ACK_count': '97839',
                     'tcp_CWR_count': '0',
                     'tcp_ECN_count': '0',
                     'tcp_FIN_count': '97839',
                     'tcp_PSH_count': '97839',
                     'tcp_RST_count': '0',
                     'tcp_SYN_count': '97839',
                     'tcp_URG_count': '0'}},
< … OMISSIS … >
 {'Relationship_3': {'City': '',
                     'CountryCode': '',
                     'ICMPTrafficBytes': '0',
                     'ICMPTrafficPackets': '0',
                     'IPv4TrafficBytes': '1200',
                     'IPv4TrafficPackets': '8',
                     'IPv6TrafficBytes': '0',
                     'IPv6TrafficPackets': '6',
                     'Organization': 'SXT2',
                     'OtherTrafficBytes': '0',
                     'OtherTrafficPackets': '0',
                     'RegionCode': '',
                     'RegionName': '',
                     'TCPTrafficBytes': '0',
                     'TCPTrafficPackets': '0',
                     'TargetAddress': '10.1.70.1',
                     'TargetFQDN': 'provincia.ip6net.me',
                     'UDPTrafficBytes': '1200',
                     'UDPTrafficPackets': '8',
                     'bytesTrafficAuthentication': '0',
                     'bytesTrafficDataStorage': '0',
                     'bytesTrafficDatabase': '0',
                     'bytesTrafficGaming': '0',
                     'bytesTrafficMAIL': '0',
                     'bytesTrafficManagement': '1200',
                     'bytesTrafficNetworkOperation': '0',
                     'bytesTrafficNetworkTunnel': '0',
                     'bytesTrafficP2P': '0',
                     'bytesTrafficPrinting': '0',
                     'bytesTrafficSocial': '0',
                     'bytesTrafficSystemsOperation': '0',
                     'bytesTrafficUnclassified': '0',
                     'bytesTrafficVPN': '0',
                     'bytesTrafficVoiceVideo': '0',
                     'bytesTrafficWEB': '0',
                     'packetsTrafficAuthentication': '0',
                     'packetsTrafficDataStorage': '0',
                     'packetsTrafficDatabase': '0',
                     'packetsTrafficGaming': '0',
                     'packetsTrafficMAIL': '0',
                     'packetsTrafficManagement': '8',
                     'packetsTrafficNetworkOperation': '0',
                     'packetsTrafficNetworkTunnel': '0',
                     'packetsTrafficP2P': '0',
                     'packetsTrafficPrinting': '0',
                     'packetsTrafficSocial': '0',
                     'packetsTrafficSystemsOperation': '0',
                     'packetsTrafficUnclassified': '0',
                     'packetsTrafficVPN': '0',
                     'packetsTrafficVoiceVideo': '0',
                     'packetsTrafficWEB': '0',
                     'tcp_ACK_count': '0',
                     'tcp_CWR_count': '0',
                     'tcp_ECN_count': '0',
                     'tcp_FIN_count': '0',
                     'tcp_PSH_count': '0',
                     'tcp_RST_count': '0',
                     'tcp_SYN_count': '0',
                     'tcp_URG_count': '0'}}]
```

def GetIPCache4Internal(user, password, server, port): This function returns the information about only the internal IPv4 hosts (as defined in the “networks” section of the */opt/fl0wer/etc/fl0wer.conf* configuration file).

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'IPCache4_entry_0': {'Areacode': '0',
                       'City': '',
                       'CountryCode': '',
                       'FQDN': 'vr10.ip6net.me',
                       'ICMPTrafficBytes': '0',
                       'ICMPTrafficPackets': '0',
                       'IGMPTrafficBytes': '0',
                       'IGMPTrafficPackets': '0',
                       'IPv4TrafficBytes': '1176012',
                       'IPv4TrafficPackets': '8142',
                       'IPv6TrafficBytes': '0',
                       'IPv6TrafficPackets': '0',
                       'ISP_AS': '',
                       'Latitude': '0.000000',
                       'Longitude': '0.000000',
                       'OtherTrafficBytes': '0',
                       'OtherTrafficPackets': '0',
                       'PostalCode': '',
                       'Region': '',
                       'Region_Name': '',
                       'TCPTrafficBytes': '0',
                       'TCPTrafficPackets': '0',
                       'Timezone': '',
                       'UDPTrafficBytes': '1176012',
                       'UDPTrafficPackets': '8142',
                       'assigned_netmask': '0',
                       'bytesTrafficAuthentication': '0',
                       'bytesTrafficDataStorage': '0',
                       'bytesTrafficDatabase': '0',
                       'bytesTrafficGaming': '0',
                       'bytesTrafficMAIL': '0',
                       'bytesTrafficManagement': '0',
                       'bytesTrafficNetworkOperation': '1176012',
                       'bytesTrafficNetworkTunnel': '0',
                       'bytesTrafficP2P': '0',
                       'bytesTrafficPrinting': '0',
                       'bytesTrafficSocial': '0',
                       'bytesTrafficSystemsOperation': '0',
                       'bytesTrafficUnclassified': '0',
                       'bytesTrafficVPN': '0',
                       'bytesTrafficVoiceVideo': '0',
                       'bytesTrafficWEB': '0',
                       'dst_bytes': '0',
                       'dst_hits': '0',
                       'dst_packets': '0',
                       'firstICMPPacketTime': '01/01/1970 01:00:00',
                       'firstIGMPPacketTime': '01/01/1970 01:00:00',
                       'firstOtherPacketTime': '01/01/1970 01:00:00',
                       'firstPacketTime': '20/03/2017 00:02:50',
                       'firstTCPPacketTime': '01/01/1970 01:00:00',
                       'firstUDPPacketTime': '20/03/2017 00:02:50',
                       'ipv4address': '10.1.10.20',
                       'lastICMPPacketTime': '01/01/1970 01:00:00',
                       'lastIGMPPacketTime': '01/01/1970 01:00:00',
                       'lastOtherPacketTime': '01/01/1970 01:00:00',
                       'lastPacketTime': '20/03/2017 18:40:37',
                       'lastTCPPacketTime': '01/01/1970 01:00:00',
                       'lastUDPPacketTime': '20/03/2017 18:40:37',
                       'organization': '',
                       'src_bytes': '1176012',
                       'src_hits': '3639',
                       'src_packets': '8142',
                       'subnet_beginip': '',
                       'subnet_endip': '',
                       'tcp_ACK_count': '0',
                       'tcp_CWR_count': '0',
                       'tcp_ECN_count': '0',
                       'tcp_FIN_count': '0',
                       'tcp_PSH_count': '0',
                       'tcp_RST_count': '0',
                       'tcp_SYN_count': '0',
                       'tcp_URG_count': '0',
                       'trafficAuthentication': '0',
                       'trafficDataStorage': '0',
                       'trafficDatabase': '0',
                          'trafficGaming': '0',
                          'trafficMAIL': '0',
                          'trafficManagement': '0',
                          'trafficNetworkOperation': '8142',
                          'trafficNetworkTunnel': '0',
                          'trafficP2P': '0',
                          'trafficPrinting': '0',
                          'trafficSocial': '0',
                          'trafficSystemsOperation': '0',
                          'trafficUnclassified': '0',
                          'trafficVPN': '0',
                          'trafficVoiceVideo': '0',
                          'trafficWEB': '0'}},
< … OMISSIS … >
 {'IPCache4_entry_71': {'Areacode': '0',
                           'City': '',
                           'CountryCode': '',
                           'FQDN': 'vrrp.mcast.net',
                           'ICMPTrafficBytes': '0',
                           'ICMPTrafficPackets': '0',
                           'IGMPTrafficBytes': '265358',
                           'IGMPTrafficPackets': '8292',
                           'IPv4TrafficBytes': '20159230',
                           'IPv4TrafficPackets': '275095',
                           'IPv6TrafficBytes': '0',
                           'IPv6TrafficPackets': '0',
                           'ISP_AS': '',
                           'Latitude': '0.000000',
                           'Longitude': '0.000000',
                           'OtherTrafficBytes': '19893872',
                           'OtherTrafficPackets': '266803',
                           'PostalCode': '',
                           'Region': '',
                           'Region_Name': '',
                           'TCPTrafficBytes': '0',
                           'TCPTrafficPackets': '0',
                           'Timezone': '',
                           'UDPTrafficBytes': '0',
                           'UDPTrafficPackets': '0',
                           'assigned_netmask': '0',
                           'bytesTrafficAuthentication': '0',
                           'bytesTrafficDataStorage': '0',
                           'bytesTrafficDatabase': '0',
                           'bytesTrafficGaming': '0',
                           'bytesTrafficMAIL': '0',
                           'bytesTrafficManagement': '0',
                           'bytesTrafficNetworkOperation': '20159230',
                           'bytesTrafficNetworkTunnel': '0',
                           'bytesTrafficP2P': '0',
                           'bytesTrafficPrinting': '0',
                           'bytesTrafficSocial': '0',
                           'bytesTrafficSystemsOperation': '0',
                           'bytesTrafficUnclassified': '0',
                           'bytesTrafficVPN': '0',
                           'bytesTrafficVoiceVideo': '0',
                           'bytesTrafficWEB': '0',
                           'dst_bytes': '20159230',
                           'dst_hits': '6716',
                           'dst_packets': '275095',
                           'firstICMPPacketTime': '01/01/1970 01:00:00',
                           'firstIGMPPacketTime': '20/03/2017 02:29:51',
                           'firstOtherPacketTime': '20/03/2017 00:01:09',
                           'firstPacketTime': '20/03/2017 00:01:09',
                           'firstTCPPacketTime': '01/01/1970 01:00:00',
                           'firstUDPPacketTime': '01/01/1970 01:00:00',
                           'ipv4address': '224.0.0.18',
                           'lastICMPPacketTime': '01/01/1970 01:00:00',
                           'lastIGMPPacketTime': '20/03/2017 18:39:17',
                           'lastOtherPacketTime': '20/03/2017 18:40:01',
                           'lastPacketTime': '20/03/2017 18:40:01',
                           'lastTCPPacketTime': '01/01/1970 01:00:00',
                           'lastUDPPacketTime': '01/01/1970 01:00:00',
                           'organization': '',
                           'src_bytes': '0',
                           'src_hits': '0',
                           'src_packets': '0',
                           'subnet_beginip': '',
                           'subnet_endip': '',
                           'tcp_ACK_count': '0',
                           'tcp_CWR_count': '0',
                           'tcp_ECN_count': '0',
                           'tcp_FIN_count': '0',
                           'tcp_PSH_count': '0',
                           'tcp_RST_count': '0',
                           'tcp_SYN_count': '0',
                           'tcp_URG_count': '0',
                           'trafficAuthentication': '0',
                           'trafficDataStorage': '0',
                           'trafficDatabase': '0',
                           'trafficGaming': '0',
                           'trafficMAIL': '0',
                           'trafficManagement': '0',
                           'trafficNetworkOperation': '275095',
                           'trafficNetworkTunnel': '0',
                           'trafficP2P': '0',
                           'trafficPrinting': '0',
                           'trafficSocial': '0',
                           'trafficSystemsOperation': '0',
                           'trafficUnclassified': '0',
                           'trafficVPN': '0',
                           'trafficVoiceVideo': '0',
                           'trafficWEB': '0'}},
    {'IPCache4_entry_EOF': {'IPCache4_entry_EOF': 'EOF'}}]
```

def GetIPCache6Internal(user, password, server, port): This function returns the information about only the internal IPv6 hosts (as defined in the “networks” section of the */opt/fl0wer/etc/fl0wer.conf* configuration file).

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'IPCache6_entry_0': {'Areacode': '0',
                        'City': '',
                        'CountryCode': '',
                        'FQDN': 'routerv6.ip6net.me',
                        'ICMPTrafficBytes': '0',
                        'ICMPTrafficPackets': '0',
                        'IGMPTrafficBytes': '0',
                        'IGMPTrafficPackets': '0',
                        'IPv4TrafficBytes': '0',
                        'IPv4TrafficPackets': '0',
                        'IPv6TrafficBytes': '0',
                        'IPv6TrafficPackets': '0',
                        'ISP_AS': '',
                        'Latitude': '0.000000',
                        'Longitude': '0.000000',
                        'OtherTrafficBytes': '0',
                        'OtherTrafficPackets': '0',
                        'PostalCode': '',
                        'Region': '',
                        'Region_Name': '',
                        'TCPTrafficBytes': '0',
                        'TCPTrafficPackets': '0',
                        'Timezone': '',
                        'UDPTrafficBytes': '0',
                        'UDPTrafficPackets': '0',
                        'assigned_netmask': '0',
                        'bytesTrafficAuthentication': '0',
                        'bytesTrafficDataStorage': '0',
                        'bytesTrafficDatabase': '0',
                        'bytesTrafficGaming': '0',
                        'bytesTrafficMAIL': '0',
                        'bytesTrafficManagement': '0',
                        'bytesTrafficNetworkOperation': '0',
                        'bytesTrafficNetworkTunnel': '0',
                        'bytesTrafficP2P': '0',
                        'bytesTrafficPrinting': '0',
                        'bytesTrafficSocial': '0',
                        'bytesTrafficSystemsOperation': '0',
                        'bytesTrafficUnclassified': '0',
                        'bytesTrafficVPN': '0',
                        'bytesTrafficVoiceVideo': '0',
                        'bytesTrafficWEB': '0',
                        'dst_bytes': '0',
                       'dst_hits': '0',
                       'dst_packets': '0',
                       'firstICMPPacketTime': '01/01/1970 01:00:00',
                       'firstIGMPPacketTime': '01/01/1970 01:00:00',
                       'firstOtherPacketTime': '01/01/1970 01:00:00',
                       'firstPacketTime': '01/01/1970 01:00:00',
                       'firstTCPPacketTime': '01/01/1970 01:00:00',
                       'firstUDPPacketTime': '01/01/1970 01:00:00',
                       'ipv6address': '2001:470:8b5b:0:10:1:30:20',
                       'lastICMPPacketTime': '01/01/1970 01:00:00',
                       'lastIGMPPacketTime': '01/01/1970 01:00:00',
                       'lastOtherPacketTime': '01/01/1970 01:00:00',
                       'lastPacketTime': '01/01/1970 01:00:00',
                       'lastTCPPacketTime': '01/01/1970 01:00:00',
                       'lastUDPPacketTime': '01/01/1970 01:00:00',
                       'organization': '',
                       'src_bytes': '72',
                       'src_hits': '1',
                       'src_packets': '1',
                       'subnet_beginip': '',
                       'subnet_endip': '',
                       'tcp_ACK_count': '0',
                       'tcp_CWR_count': '0',
                       'tcp_ECN_count': '0',
                       'tcp_FIN_count': '0',
                       'tcp_PSH_count': '0',
                       'tcp_RST_count': '0',
                       'tcp_SYN_count': '0',
                       'tcp_URG_count': '0',
                       'trafficAuthentication': '0',
                       'trafficDataStorage': '0',
                       'trafficDatabase': '0',
                       'trafficGaming': '0',
                       'trafficMAIL': '0',
                       'trafficManagement': '0',
                       'trafficNetworkOperation': '0',
                       'trafficNetworkTunnel': '0',
                       'trafficP2P': '0',
                       'trafficPrinting': '0',
                       'trafficSocial': '0',
                       'trafficSystemsOperation': '0',
                       'trafficUnclassified': '0',
                       'trafficVPN': '0',
                       'trafficVoiceVideo': '0',
                       'trafficWEB': '0'}},
< … OMISSIS … >
 {'IPCache6_entry_8': {'Areacode': '0',
                       'City': '',
                       'CountryCode': '',
                       'FQDN': '2001:470:8b5b:0:10:100:3:20',
                       'ICMPTrafficBytes': '0',
                       'ICMPTrafficPackets': '0',
                       'IGMPTrafficBytes': '0',
                       'IGMPTrafficPackets': '0',
                       'IPv4TrafficBytes': '0',
                       'IPv4TrafficPackets': '0',
                       'IPv6TrafficBytes': '5760',
                       'IPv6TrafficPackets': '45',
                       'ISP_AS': '',
                       'Latitude': '0.000000',
                       'Longitude': '0.000000',
                       'OtherTrafficBytes': '5760',
                       'OtherTrafficPackets': '45',
                       'PostalCode': '',
                       'Region': '',
                       'Region_Name': '',
                       'TCPTrafficBytes': '0',
                       'TCPTrafficPackets': '0',
                       'Timezone': '',
                       'UDPTrafficBytes': '0',
                       'UDPTrafficPackets': '0',
                       'assigned_netmask': '0',
                       'bytesTrafficAuthentication': '0',
                       'bytesTrafficDataStorage': '0',
                       'bytesTrafficDatabase': '0',
                       'bytesTrafficGaming': '0',
                       'bytesTrafficMAIL': '0',
                       'bytesTrafficManagement': '0',
                     'bytesTrafficNetworkOperation': '5760',
                     'bytesTrafficNetworkTunnel': '0',
                     'bytesTrafficP2P': '0',
                     'bytesTrafficPrinting': '0',
                     'bytesTrafficSocial': '0',
                     'bytesTrafficSystemsOperation': '0',
                     'bytesTrafficUnclassified': '0',
                     'bytesTrafficVPN': '0',
                     'bytesTrafficVoiceVideo': '0',
                     'bytesTrafficWEB': '0',
                     'dst_bytes': '0',
                     'dst_hits': '0',
                     'dst_packets': '0',
                     'firstICMPPacketTime': '01/01/1970 01:00:00',
                     'firstIGMPPacketTime': '01/01/1970 01:00:00',
                     'firstOtherPacketTime': '20/03/2017 18:17:32',
                     'firstPacketTime': '20/03/2017 18:17:32',
                     'firstTCPPacketTime': '01/01/1970 01:00:00',
                     'firstUDPPacketTime': '01/01/1970 01:00:00',
                     'ipv6address': '2001:470:8b5b:0:10:100:3:20',
                     'lastICMPPacketTime': '01/01/1970 01:00:00',
                     'lastIGMPPacketTime': '01/01/1970 01:00:00',
                     'lastOtherPacketTime': '20/03/2017 18:40:57',
                     'lastPacketTime': '20/03/2017 18:40:57',
                     'lastTCPPacketTime': '01/01/1970 01:00:00',
                     'lastUDPPacketTime': '01/01/1970 01:00:00',
                     'organization': '',
                     'src_bytes': '6144',
                     'src_hits': '11',
                     'src_packets': '48',
                     'subnet_beginip': '',
                     'subnet_endip': '',
                     'tcp_ACK_count': '0',
                     'tcp_CWR_count': '0',
                     'tcp_ECN_count': '0',
                     'tcp_FIN_count': '0',
                     'tcp_PSH_count': '0',
                     'tcp_RST_count': '0',
                     'tcp_SYN_count': '0',
                     'tcp_URG_count': '0',
                     'trafficAuthentication': '0',
                     'trafficDataStorage': '0',
                     'trafficDatabase': '0',
                     'trafficGaming': '0',
                     'trafficMAIL': '0',
                     'trafficManagement': '0',
                     'trafficNetworkOperation': '45',
                     'trafficNetworkTunnel': '0',
                     'trafficP2P': '0',
                     'trafficPrinting': '0',
                     'trafficSocial': '0',
                     'trafficSystemsOperation': '0',
                     'trafficUnclassified': '0',
                     'trafficVPN': '0',
                     'trafficVoiceVideo': '0',
                     'trafficWEB': '0'}},
 {'IPCache6_entry_EOF': {'IPCache6_entry_EOF': 'EOF'}}]
```

Network Protocol functions

def GetProtocolsByHits(user, password, server, port): This function returns the top 100 protocols sorted by absolute flows count. First are returned the TCP protocols, then the UDP protocols and at the end the other protocols.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'tcpStat0': {'bytes': '899127781',
               'description': 'TCP 443 HTTPS/SSLVPN Protocol',
               'hits': '292333',
               'packets': '1421400',
               'port': '443',
               'protocol': '6'}},
 {'tcpStat1': {'bytes': '4713740',
               'description': 'TCP 6080 Ubiquiti mFI Server',
               'hits': '56562',
               'packets': '96143',
               'port': '6080',
               'protocol': '6'}},
< … OMISSIS … >
 {'tcpStat99': {'bytes': '105325490',
               'description': 'TCP 80 HTTP Protocol',
               'hits': '39386',
               'packets': '201012',
               'port': '80',
               'protocol': '6'}},
 {'udpStat0': {'bytes': '406054429',
               'description': 'UDP 53 DNS Protocol',
               'hits': '2764630',
               'packets': '4591992',
               'port': '53',
               'protocol': '17'}},
 {'udpStat1': {'bytes': '4219144845',
               'description': 'UDP 2056 Cisco Netflow/IPFIX Protocol',
               'hits': '242625',
               'packets': '4581224',
               'port': '2056',
               'protocol': '17'}},
 {'udpStat99': {'bytes': '28714106',
               'description': 'UDP 161 SNMP Protocol',
               'hits': '239246',
               'packets': '360433',
               'port': '161',
               'protocol': '17'}},
< … OMISSIS … >
```

```
 {'othStat1': {'bytes': '1511941990',
              'description': 'TCP',
              'hits': '506887',
              'packets': '2639060',
              'port': '0',
              'protocol': '6'}},
 {'othStat2': {'bytes': '364087237',
              'description': 'ICMP',
              'hits': '273011',
              'packets': '3922218',
              'port': '0',
              'protocol': '1'}},
< … OMISSIS … >
 {'othStat99': {'bytes': '0',
               'description': 'ENCAP',
               'hits': '0',
               'packets': '0',
               'port': '0',
               'protocol': '98'}}]
```

def GetProtocolsByPackets(user, password, server, port): This function returns the top 100 protocols sorted by absolute packet count. First are returned the TCP protocols, then the UDP protocols and at the end the other protocols.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'tcpStat0': {'bytes': '899127781',
              'description': 'TCP 443 HTTPS/SSLVPN Protocol',
              'hits': '292333',
              'packets': '1421400',
              'port': '443',
              'protocol': '6'}},
 {'tcpStat1': {'bytes': '4713740',
              'description': 'TCP 6080 Ubiquiti mFI Server',
              'hits': '56562',
              'packets': '96143',
              'port': '6080',
              'protocol': '6'}},
< … OMISSIS … >
 {'tcpStat99': {'bytes': '105325490',
              'description': 'TCP 80 HTTP Protocol',
              'hits': '39386',
              'packets': '201012',
              'port': '80',
              'protocol': '6'}},
 {'udpStat0': {'bytes': '406054429',
              'description': 'UDP 53 DNS Protocol',
              'hits': '2764630',
              'packets': '4591992',
              'port': '53',
              'protocol': '17'}},
 {'udpStat1': {'bytes': '4219144845',
              'description': 'UDP 2056 Cisco Netflow/IPFIX Protocol',
              'hits': '242625',
              'packets': '4581224',
              'port': '2056',
              'protocol': '17'}},
 {'udpStat99': {'bytes': '28714106',
              'description': 'UDP 161 SNMP Protocol',
              'hits': '239246',
              'packets': '360433',
              'port': '161',
              'protocol': '17'}},
< … OMISSIS … >
```

```
 {'othStat1': {'bytes': '1511941990',
               'description': 'TCP',
               'hits': '506887',
               'packets': '2639060',
               'port': '0',
               'protocol': '6'}},
 {'othStat2': {'bytes': '364087237',
               'description': 'ICMP',
               'hits': '273011',
               'packets': '3922218',
               'port': '0',
               'protocol': '1'}},
< … OMISSIS … >
 {'othStat99': {'bytes': '0',
                'description': 'ENCAP',
                'hits': '0',
                'packets': '0',
                'port': '0',
                'protocol': '98'}}]
```

def GetProtocolsByBytes(user, password, server, port): This function returns the top 100 protocols sorted by absolute byte count. First are returned the TCP protocols, then the UDP protocols and at the end the other protocols.

In case of successful query, the following JSON fields are returned (values are example data):

```
[{'tcpStat0': {'bytes': '89912231237781',
               'description': 'TCP 443 HTTPS/SSLVPN Protocol',
               'hits': '292333',
               'packets': '1421400',
               'port': '443',
               'protocol': '6'}},
 {'tcpStat1': {'bytes': '4712312313740',
               'description': 'TCP 6080 Ubiquiti mFI Server',
               'hits': '56562',
               'packets': '96143',
               'port': '6080',
               'protocol': '6'}},
< … OMISSIS … >
 {'tcpStat99': {'bytes': '105325490',
               'description': 'TCP 80 HTTP Protocol',
               'hits': '39386',
               'packets': '201012',
               'port': '80',
               'protocol': '6'}},
 {'udpStat0': {'bytes': '401321236054429',
               'description': 'UDP 53 DNS Protocol',
               'hits': '2764630',
               'packets': '4591992',
               'port': '53',
               'protocol': '17'}},
 {'udpStat1': {'bytes': '4219213144845',
               'description': 'UDP 2056 Cisco Netflow/IPFIX Protocol',
               'hits': '242625',
               'packets': '4581224',
               'port': '2056',
               'protocol': '17'}},
 {'udpStat99': {'bytes': '28714106',
               'description': 'UDP 161 SNMP Protocol',
               'hits': '239246',
               'packets': '360433',
               'port': '161',
               'protocol': '17'}},
< … OMISSIS … >
```

```
 {'othStat1': {'bytes': '1511231231941990',
               'description': 'TCP',
               'hits': '506887',
               'packets': '2639060',
               'port': '0',
               'protocol': '6'}},
 {'othStat2': {'bytes': '36401233287237',
               'description': 'ICMP',
               'hits': '273011',
               'packets': '3922218',
               'port': '0',
               'protocol': '1'}},
< … OMISSIS … >
 {'othStat99': {'bytes': '0',
                'description': 'ENCAP',
                'hits': '0',
                'packets': '0',
                'port': '0',
                'protocol': '98'}}]
```

Reconfiguration functions

These functions normally don’t return data or result codes since their action is executed immediately on the Fl0wer server.

def ReloadCustomNetworks(user, password, server, port, ip): Dynamically reloads the Custom Networks List on the Fl0wer server.

def EnableIPCache(user, password, server, port): Dynamically enable the IP Cache feature on the Fl0wer server.

def DisableIPCache(user, password, server, port): Dynamically disable the IP Cache feature on the Fl0wer server.

def EnableNPAR(user, password, server, port): Dynamically enable the NPAR feature on the Fl0wer server.

def DisableNPAR(user, password, server, port): Dynamically disable the NPAR feature on the Fl0wer server.

def EnableNTPList(user, password, server, port): Dynamically enable the NTP Lists on the Fl0wer server.

def DisableNTPList(user, password, server, port): Dynamically disable the NTP Lists on the Fl0wer server.

def EnableBGPList(user, password, server, port): Dynamically enable the BGP Lists on the Fl0wer server.

def DisableBGPList(user, password, server, port): Dynamically disable the BGP Lists on the Fl0wer server.

def EnableDNSList(user, password, server, port): Dynamically enable the DNS Lists on the Fl0wer server.

def DisableDNSList(user, password, server, port): Dynamically disable the DNS Lists on the Fl0wer server.

def EnableTORList(user, password, server, port): Dynamically enable the TOR List on the Fl0wer server.

def DisableTORList(user, password, server, port): Dynamically disable the TOR List on the Fl0wer server.

def EnableCustomNetworks(user, password, server, port): Dynamically enable the Custom Networks on the Fl0wer server.

def DisableCustomNetworks(user, password, server, port): Dynamically disable the Custom Networks on the Fl0wer server.

def EnableTrafficRules(user, password, server, port): Dynamically enable the Traffic Rules on the Fl0wer server.

def DisableTrafficRules(user, password, server, port): Dynamically disable the Traffic Rules on the Fl0wer server.

Top 100 Flows Functions

These functions return the top 100 flows sorted by size, packets or duration.

def GetTop100FlowsBySize(user, password, server, port): Returns the top 100 flows sorted by size (in bytes)since the beginning of data collection.

def GetTop100FlowsByPackets(user, password, server, port): Returns the top 100 flows sorted by number of packets since the beginning of data collection.

def GetTop100FlowsByDuration(user, password, server, port): Returns the top 100 flows sorted by seconds duration since the beginning of data collection.

def GetTopV4Relations(user, password, server, port): Returns the top 100 IPv4 Hosts sorted by number of different relationships

def GetTopV6Relations(user, password, server, port): Returns the top 100 IPv6 Hosts sorted by number of different relationships

General Network Security Related Functions

These functions return the last 1000 flows related to several security areas.

def GetLastFlowsBogons(user, password, server, port): Returns the last 1000 flows involving traffic with Network Bogons (IP Prefixes that should not be available on the Internet). Both IPv4 and IPv6 flows are returned.

def GetLastFlowsReputation(user, password, server, port): Returns the last 1000 flows involving traffic with IP Addresses or Prefixes listed in the Bad Reputation IP List. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsTOR(user, password, server, port): Returns the last 1000 flows involving traffic with TOR nodes (identified by lists or by NPAR). Both IPv4 and IPv6 flows are returned.

def GetLastFlowsP2P(user, password, server, port): Returns the last 1000 flows involving P2P traffic as detected by NPAR. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsRisky(user, password, server, port): Returns the last 1000 flows involving clear-text traffic going to or from the Internet. Both IPv4 and IPv6 flows are returned. Note: HTTP, QUIC and DNS traffic are not listed here.

def GetLastFlowsUnclassified(user, password, server, port): Returns the last 1000 flows involving traffic that could not be classified by NPAR. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsVPN(user, password, server, port): Returns the last 1000 flows involving VPN traffic classified by NPAR to and from the Internet. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsAuthentication(user, password, server, port): Returns the last 1000 flows involving Authentication traffic classified by NPAR to and from the Internet. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsTunnel(user, password, def GetLastFlowsReputation(user, password, server, port):server, port): Returns the last 1000 flows involving Network Tunnel traffic classified by NPAR to and from the Internet. Both IPv4 and IPv6 flows are returned.

def GetLastFlowsExtMgmt(user, password, server, port): Returns the last 1000 flows involving System Management traffic classified by NPAR to and from the Internet. Both IPv4 and IPv6 flows are returned.

def GetUnknownDNS(user, password, server, port): Returns the list of addresses doing DNS traffic that are not listed in the relative list file. Both IPv4 and IPv6 addresses are returned.

def GetUnknownNTP(user, password, server, port): Returns the list of addresses doing NTP traffic that are not listed in the relative list file. Both IPv4 and IPv6 addresses are returned.

def GetUnknownBGP(user, password, server, port): Returns the list of addresses doing BGP traffic that are not listed in the relative list file. Both IPv4 and IPv6 addresses are returned.

Network Scans Related Functions

These functions return the list of scanners, targets and the last 1000 flows involving network scanning. Currently the scan detection module detects SYN, NULL, FIN and XMAS TCP scans, both for IPv4 and IPv6. The UDP scan module is still in development.

def GetScanners(user, password, server, port): Returns the list of addresses of IPs doing network scans. Note: Due to the nature of how some scans or floods works, it is possible that replies of the targets or victims are considered as sources or destinations of the scan or flood. Despite that, indagation should be performed on traffic or hosts since this is a good alarm bell.

def GetTargets(user, password, server, port): Returns the list of addresses of IPs being targeted by network scans. Note: Due to the nature of how some scans or floods works, it is possible that replies of the targets or victims are considered as sources or destinations of the scan or flood. Despite that, indagation should be performed on traffic or hosts since this is a good alarm bell.

def GetLastScans(user, password, server, port): Returns the last 1000 flows involving network scans.

Network Floods Related Functions

def GetFlooders(user, password, server, port): Returns the list of addresses of IPs doing network floods. Note: Due to the nature of how some scans or floods works, it is possible that replies of the targets or victims are considered as sources or destinations of the scan or flood. Despite that, indagation should be performed on traffic or hosts since this is a good alarm bell.

def GetFloodVictims(user, password, server, port): Returns the list of addresses of IPs being targeted by network floods. Note: Due to the nature of how some scans or floods works, it is possible that replies of the targets or victims are considered as sources or destinations of the scan or flood. Despite that, indagation should be performed on traffic or hosts since this is a good alarm bell.

def GetLastFloods(user, password, server, port): Returns the last 1000 flows involving network floods.

The CLI utilities

The CLI utilities are divided in two main categories:

1. the tools operating on the flw raw binary files (those with the flw- prefix)
2. the tools interacting with the fl0wer daemon (those with the flcli- prefix)

While full sources are provided for both categories, these utilities can be very helpful and not used only as platform programming examples but as daily automation instruments.

Binary File Management

The flw utilities are written in C language for performance reasons and provide the following functions:

flw-csv: this tool reads a .flw binary file and outputs the equivalent .CSV (Comma Separated Value) file, suitable for use in shell scripts and importing into spreadsheet programs. It has several flags that allows the user to filter by IP Address, Flow Exporter, NPAR and source or destination port.

/opt/fl0wer/bin/flw-json: this tool reads a .flw binary file and outputs the equivalent JSON file.

/opt/fl0wer/bin/flw-sqlite: this tool reads the .flw binary file and outputs a DML (Data Management Language) suitable to be imported into a sqlite3 database. This is used by one of the flcli tools to report on the flow-matrix used by an IP address.

/opt/fl0wer/bin/flw-oracle: this tool reads the .flw binary file and outputs a DML (Data Management Language) suitable to be imported into an Oracle database.

/opt/fl0wer/bin/flw-timespan: this simple tool shows you the first and last flow timestamp so the user can easily understand the time frame coverage of the .flw file.

Daemon Configuration

The flcli utilities are written in Python3 for simplicity, normally use the JSON API where needed and provide the following functions:

/opt/fl0wer/bin/flcli-mkpemcert: this tool is a quick'n dirty hack to create a .pem file for the TLSv1.2 encryption to be used by the Fl0wer daemon when listening for requests on the TCP/7443 port. It is used at the install to provide the user with a ready to run Fl0wer daemon.

User Management

The following set of commands allow the Fl0wer administrator to manage the user database.

/opt/fl0wer/bin/flcli-createusrdb: this tool creates and initializes an empty user database. The user database is located in the */opt/fl0wer/etc/fl0werusers.db* file and cannot be moved from there. It is a simple SQLite 3 table with 3 fields (only the username and password are currently used now).
The passwords are stored as SHA-512 digests.
When executed (like in the install package phase), a default user is created:

```
user:           admin
password:       fl0werr0x
```

►It is strongly suggested to change the password **immediately** using the */opt/fl0wer/bin/flcli-setuserpw* command, or even better, add your users and remove the admin name if not required.

/opt/fl0wer/bin/flcli-adduser: this command allows the administrator to add a new user into the fl0werusers.db database, specifying the username and the password, which will be encrypted.

/opt/fl0wer/bin/flcli-deluser: this tool removes the user from the fl0werusers.db database, specifying the username.

/opt/fl0wer/bin/flcli-lsuserdb: this tool dumps the user names and the SHA-512 digests stored in the SQLite3 */opt/fl0wer/etc/fl0werusers.db* database.

/opt/fl0wer/bin/flcli-setuserpw: this tool allows a manager to change the password for a specified username.

Network Information & Reporting

The following commands interact with the Fl0wer daemon using the JSON API on the TCP/7443 port. They all have the same parameters (the parameters are: <username> <password> <fl0wer_server> <tcp_port>)
and return the described data using a tabular format, suitable for use in shell scripting.

/opt/fl0wer/bin/flcli-rtgetexpstatsv4: returns a list of all the IPv4 Flow Exporters with the data they’ve seen from the last cleanup. All values are absolute.

/opt/fl0wer/bin/flcli-rtgetexpstatsv6: returns a list of all the IPv6 Flow Exporters with the data they’ve seen from the last cleanup. All values are absolute.

/opt/fl0wer/bin/flcli-rtgetflowsall: returns a list of the last N flows (as per the *flows_to_keep_in_ram* configuration file parameter). The flows returned by this function are all the IPv4, Ipv6 and other flows.

/opt/fl0wer/bin/flcli-rtgetflowsoth: returns a list of the last N Other (non TCP and not UDP) flows (as per the *flows_to_keep_in_ram* configuration file parameter). The flows returned by this function are all the IPv4, Ipv6 and other flows.

/opt/fl0wer/bin/flcli-rtgetflowsv4: returns a list of the last N IPv4 flows (as per the *flows_to_keep_in_ram* configuration file parameter).

/opt/fl0wer/bin/flcli-rtgetflowsv6: returns a list of the last N IPv6 flows (as per the *flows_to_keep_in_ram* configuration file parameter).

/opt/fl0wer/bin/flcli-rtgethostsv4: returns a list of all the hosts known in the IPv4 Cache.
Can be a very big table.

/opt/fl0wer/bin/flcli-rtgethostsv6: returns a list of all the hosts known in the IPv6 Cache.
Can be a very big table.

/opt/fl0wer/bin/flcli-rtgetprotostats: this tool returns a sorted list (by bytes) of the top 100 TCP, UDP and other protocols in a tabular form.
/opt/fl0wer/bin/flcli-rtgettemplates: this tool returns in a tabular form all the templates received by Fl0wer, including all the fields details.

/opt/fl0wer/bin/flcli-ping: this command returns basic information about a Fl0wer daemon. It is suitable to be used if daemon is alive when using a monitoring tool like Nagios or Icinga.

/opt/fl0wer/bin/flcli-snapshot: this command creates a snapshot file from a running Fl0wer server storing the retrieved information in a zip file named with a syntax like: fl0wer-netsnap-17-3-2017_2-31-41.zip (fl0wer-netsnap-DD-MM-YYYY-HH-MM-SS.zip)

The following commands interact with the Fl0wer daemon using the JSON API over the TCP/7443 port. They all have the same parameters (the parameters are: <username> <password> <fl0wer_server> <tcp_port> <ipaddress>) and return the described data using JSON Notation.

/opt/fl0wer/bin/flcli-rtgethostinfo: returns the information about a specific IP address from the relative IP Cache using JSON notation.

/opt/fl0wer/bin/flcli-rtgetexporterinfo: returns the information about a specific Flow Exporter using its IP address using JSON notation.

/opt/fl0wer/bin/flcli-mkreport: this command is a powerful tool that allows you to obtain the traffic matrix for a specific IP Address, including bytes and packets seen, flow-direction and NPAR data if available.
It works together with the flw-sqlite tool and the sqlite3 command.
Its syntax is as follows:

```
flcli-mkreport  <database> <tablename> <ipaddress>
```

Example usage (we want to know all data regarding the IP 10.1.60.221):

It is first necessary convert the .flw file (like the /opt/fl0wer/data/netflow.flw) into an SQLite DML:

```
$ ./flw-sqlite /opt/fl0wer/data/netflow.flw /tmp/work.dml today
```

Then data can be imported into SQLite3:

```
$ sqlite3 mynewdatabase.db < /tmp/work.dml
```

And then you can use the flcli-mkreport tool:

```
$ flcli-mkreport mynewdatabase.db today 10.1.60.221
```

```
-----------------------------------------------------------------------------------------------
Network Activity report for IP:  10.1.60.221
-----------------------------------------------------------------------------------------------
+--------------------+------------------+------------------+
|  TCP Total Packets |  TCP Total Bytes |  TCP Total Flows |
+====================+==================+==================+
|             211360 |        129038934 |            38772 |
+--------------------+------------------+------------------+
+--------------------+------------------+------------------+
|  UDP Total Packets |  UDP Total Bytes |  UDP Total Flows |
+====================+==================+==================+
|              94924 |         67094870 |            10678 |
+--------------------+------------------+------------------+
+----------------------+--------------------+--------------------+
|  Other Total Packets |  Other Total Bytes |  Other Total Flows |
+======================+====================+====================+
|                 9596 |            4259896 |               3482 |
+----------------------+--------------------+--------------------+
------- Hits as Source TCP TRAFFIC --------------
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| destinationIP   |   protocol |    bytes |   destinationPort | NPAR                                        | Flow Direction       |
+=================+============+==========+===================+=============================================+======================+
| 23.227.55.32    |          6 |  5800054 |               443 | TCP 443 HTTPS/SSLVPN Protocol               | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 31.13.74.1      |          6 |  1351329 |               443 | Facebook                                    | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 10.1.30.210     |          6 |  1340590 |              7443 | TCP 7443 Fl0wer default JSON Interface      | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 10.1.30.220     |          6 |  1272156 |                22 | TCP 22 SSH/SFTP Protocol                    | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 151.101.129.69  |          6 |   559177 |                80 | Reddit Social Network                       | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 52.29.111.36    |          6 |   521276 |               443 | Amazon - Service:AMAZON eu-central-1        | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 10.1.30.220     |          6 |   363307 |                80 | TCP 80 HTTP Protocol                        | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 92.39.246.45    |          6 |   269532 |              9001 | TCP 9001 TOR Protocol                       | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 148.251.136.84  |          6 |   144108 |               143 | TCP 143 MAIL IMAPv4 Protocol                | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 104.16.112.18   |          6 |   126843 |               443 | Cloudflare IPv4 #4                          | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 64.233.166.16   |          6 |   109065 |               993 | TCP 993 MAIL IMAPv4/IMAPS/IMAP-SSL Protocol | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 74.125.71.188   |          6 |   100952 |              5228 | TCP 5228 Google Chrome/Android Playstore    | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 23.23.98.233    |          6 |    52127 |               443 | Amazon - Service:AMAZON us-east-1           | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 10.1.30.101     |          6 |    45245 |               179 | TCP 179 ROUTING BGP Protocol                | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 172.217.16.195  |          6 |    24208 |                80 | GMail IP Range                              | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
| 52.222.157.28   |          6 |     3648 |                80 | Amazon - Service:AMAZON GLOBAL              | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+---------------------------------------------+----------------------+
------- Hits as Destination TCP TRAFFIC --------------
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| sourceIP       |   protocol |    bytes |   destinationPort | NPAR                                             | Flow Direction       |
+================+============+==========+===================+==================================================+======================+
| 10.1.30.210    |          6 | 83653412 |             33145 | TCP 7443 Fl0wer default JSON Interface           | INTERNAL_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 23.227.55.32   |          6 | 20243886 |             54883 | TCP 443 HTTPS/SSLVPN Protocol                    | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 31.13.86.4     |          6 |  3539215 |             13250 | Facebook                                         | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 10.1.30.220    |          6 |  3381668 |             19725 | TCP 22 SSH/SFTP Protocol                         | INTERNAL_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 92.39.246.45   |          6 |  2169674 |             13759 | TCP 9001 TOR Protocol                            | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 151.101.1.69   |          6 |  1764856 |             29370 | Reddit Social Network                            | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 52.29.111.36   |          6 |  1043066 |             18509 | Amazon - Service:AMAZON eu-central-1             | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 104.16.112.18  |          6 |   368909 |             23336 | Cloudflare IPv4 #4                               | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 10.1.30.220    |          6 |   351353 |             59728 | TCP 80 HTTP Protocol                             | INTERNAL_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 64.233.166.16  |          6 |   118949 |             30878 | TCP 993 MAIL IMAPv4/IMAPS/IMAP-SSL Protocol      | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 23.23.98.233   |          6 |    82264 |             56452 | Amazon - Service:AMAZON us-east-1                | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+--------------------------------------------------+----------------------+
| 74.125.71.188  |          6 |    64303 |             33020 | TCP 5228 Google Chrome/Android Playstore         | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 148.251.136.84 |          6 |    64203 |             41627 | TCP 143 MAIL IMAPv4 Protocol                    | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 10.1.30.101    |          6 |    44827 |             53459 | TCP 179 ROUTING BGP Protocol                    | INTERNAL_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 172.217.16.195 |          6 |    24222 |             57886 | GMail IP Range                                  | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 192.0.73.2     |          6 |    21454 |             50000 | TCP 50000 IBM DB/2                              | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 23.2.55.167    |          6 |    17208 |             37017 | TCP 37017 Ubiquiti mFI MongoDB default instance | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
| 52.222.157.28  |          6 |     1848 |             53534 | Amazon - Service:AMAZON GLOBAL                  | INTERNET_TO_INTERNAL |
+----------------+------------+----------+-------------------+-------------------------------------------------+----------------------+
------- Hits as Source UDP TRAFFIC --------------
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| destinationIP   |   protocol |    bytes |   destinationPort | NPAR                                      | Flow Direction       |
+=================+============+==========+===================+===========================================+======================+
| 10.100.3.20     |         17 | 14971496 |              2055 | UDP 2055 Cisco Netflow/IPFIX Protocol     | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 216.58.205.206  |         17 |  3411859 |               443 | UDP 443 Google Chrome QUIC Protocol       | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 10.1.30.210     |         17 |   457379 |               514 | UDP 514 SYSLOG Protocol                   | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 239.255.255.250 |         17 |   252730 |              1900 | UDP 1900 UPnP                             | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 172.217.16.195  |         17 |    92139 |               443 | GMail IP Range                            | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 192.168.37.2    |         17 |    71118 |               137 | UDP 137 NetBIOS Name Service Protocol     | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 10.1.30.150     |         17 |    45188 |                53 | UDP 53 DNS Protocol                       | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 10.1.60.255     |         17 |    42807 |               138 | UDP 138 NetBIOS Datagram Service Protocol | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 52.29.111.36    |         17 |    37686 |               443 | Amazon - Service:AMAZON eu-central-1      | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 10.1.30.38      |         17 |    37000 |               161 | UDP 161 SNMP Protocol                     | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 224.0.0.251     |         17 |    33345 |              5353 | UDP 5353 Multicast DNS Apple Bonjour      | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
| 193.204.114.233 |         17 |    18164 |               123 | UDP 123 NTP Protocol                      | INTERNAL_TO_INTERNET |
+-----------------+------------+----------+-------------------+-------------------------------------------+----------------------+
------- Hits as Destination UDP TRAFFIC --------------
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| sourceIP        |   protocol |    bytes |   destinationPort | NPAR                                                             | Flow Direction       |
+=================+============+==========+===================+==================================================================+======================+
| 10.100.3.20     |         17 | 42065408 |              2055 | UDP 2055 Cisco Netflow/IPFIX Protocol                            | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 216.58.205.206  |         17 |  3959057 |             35095 | UDP 443 Google Chrome QUIC Protocol                              | INTERNET_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 10.1.70.1       |         17 |  1139712 |              2056 | UDP 2056 Cisco Netflow/IPFIX Protocol                            | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 172.217.16.195  |         17 |   317036 |             33388 | GMail IP Range                                                   | INTERNET_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 10.1.30.150     |         17 |   129214 |             61895 | UDP 53 DNS Protocol                                              | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 193.204.114.233 |         17 |    12844 |             12300 | UDP 123 NTP Protocol                                             | INTERNET_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 10.1.30.150     |         17 |      352 |             13060 | UDP 13060 3CX Cloud SIP Protocol                                 | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
| 10.1.30.150     |         17 |      336 |              7004 | UDP 7004 AFS Kerberos authentication service/CEPH OSD Filesystem | INTERNAL_TO_INTERNAL |
+-----------------+------------+----------+-------------------+------------------------------------------------------------------+----------------------+
------- Hits as Source Other TRAFFIC --------------
+-----------------+------------+---------+-----------+---------------------------------------------+----------------------+
| destinationIP   |   protocol |   bytes |   packets | NPAR                                        | Flow Direction       |
+=================+============+=========+===========+=============================================+======================+
| 10.100.3.20     |          1 | 2745108 |       444 | ICMP Protocol Code: Not available           | INTERNAL_TO_INTERNAL |
+-----------------+------------+---------+-----------+---------------------------------------------+----------------------+
| 10.100.3.20     |          1 | 1514788 |         1 | ICMP Protocol Code: DESTINATION UNREACHABLE | INTERNAL_TO_INTERNAL |
+-----------------+------------+---------+-----------+---------------------------------------------+----------------------+
------- Hits as Destination Other TRAFFIC --------------
+------------+------------+---------+-----------+--------+------------------+
| sourceIP   | protocol   | bytes   | packets   | NPAR   | Flow Direction   |
+============+============+=========+===========+========+==================+
+------------+------------+---------+-----------+--------+------------------+
```

The GUI Application (Fl0werUI)

Probably the first thing a Fl0wer user is going to interact with is the Fl0werUI GUI Application. Fl0werUI was born with two ambitious targets in its inception:

1. provide a portable, performing and simple to use GUI to interact with the Fl0wer Server
2. provide a portable and simple to understand coding (customizable) example to show how to use most of the Fl0wer JSON APIs

To fulfill the above said targets, the chosen platform is Python 3 with the TkInter Toolkit. Nowadays the Fl0werUI application runs in at least three environments:

- Linux (using X11)
- Windows 32/64 bit
- Mac OSX Yosemite and probably greater

The application use is pretty simple, when executed it prompts for the credentials and address to connect to a Fl0wer server. If the connection and authentication are successful, data is downloaded from the Fl0wer server using a RESTful model and presented to the user. If you skip the connection phase using the ***Skip*** button, an “empty” application is shown and it is possible, using the menus, to connect to a server or to load data from a Network Snapshot. In this way, as example, a customer could schedule Network Snapshots on the server and send them to an external consultant for analysis.

The application by itself is pretty simple, it shows lists of data using several “sorting” keys. Most (but not all) objects are clickable showing details and insights of the requested information. The last tab shows a list of Pie Charts to get a good idea of how the overall Network is performing.

Limitations

Fl0wer is a young product and is constantly developed. I have a lot of new ideas and a lot of work is being done on a daily basis, anyway it still has some limits that are being worked on. Current known limitations are:

- ➔ NBAR: it is still in an experimental phase, but not still usable. It is a pretty proprietary feature (Cisco Only) but can be useful.
- ➔ All Netflow V9/IPFIX data packets received before the relative templates are sent by the flow exporters is simply discarded.
- ➔ No PEN support for Netflow V9/IPFIX. Most vendors implementing PEN data types inside their products are not releasing public specs for how the PEN fields should be decoded and used.
- ➔ A DDoS is still missing. The Scanning module (for now) is limited to TCP (for both IPv4 and IPv6) and the Flood detection module is limited (for now) to ICMP (IPv4 and IPv6).
- ➔ All examples and open-source code are written in Python 3 for sake of portability, it would be nice to have some examples in other common programming languages.
- ➔ When the Fl0wer server daemon aborts or is killed with -9 signal (SIGKILL) it may require up to 2 minutes to restart it.

Support

The Fl0wer server daemon is supported for free on a best-effort basis using the email contact: support@Fl0wer.me . This kind of supports allows configuration information requests and reporting of bugs, but all bug resolutions (where possible) are queued to the next release of software. If it is not possible to wait for the next release, paid support can be bought contacting the info@fl0wer.me mail account.

No free support is provided on the open-source parts of the Fl0wer solution (the CLI tools, the Fl0werUI GUI application) due to the variety of platforms available on the market.

Troubleshooting

When requesting for support (free or paid), be sure to be able to provide the following information:

- tcpdump captures of the interested network side. Tcpdump command or the Wireshark tool can be used to get the requested info.

Further, the output of the following commands will be requested, so be sure to have root privileges on the system running the Fl0wer server. For your convenience, a script (to be run as root user) is provided in */opt/fl0wer/scripts/flcli-supportinfo.sh* that executes the above said commands and creates a *supportinfo.tar* file in the */tmp* directory. Depending on the issue, more command outputs could be requested.

Command to be executed	**Reason**
cat /proc/cpuinfo	To understand on which hardware is Fl0wer running.
cat /proc/mdstat	To understand storage topology.
cat /etc/security/limits.conf	To verify system configured limits.
cat /etc/sysctl.conf	To check the network stack configuration.
pstree	To have an overview of the running processes.
ps -efa	To have an overview of the running processes.
find /proc/`pgrep fl0wer` -ls	To have an overview of the running processes.
free	To have an overview of the memory status.
cat /proc/meminfo	To have an overview of the memory status.
cat /etc/hosts	To verify network resolution issues.
cat /etc/redhat-release	To check the RedHat/CentOS version, if it is in use.
cat /etc/debian_version	To check the Debian/Ubuntu version, if it is in use.

Command to be executed	**Reason**
cat /etc/passwd	To verify the correct Fl0wer user configuration.
cat /etc/group	To verify the correct Fl0wer user configuration.
getent passwd	To verify the correct Fl0wer user configuration.
getent group	To verify the correct Fl0wer user configuration.
cat /etc/nsswitch.conf	To understand the resolution mechanism in use.
find /opt/fl0wer -ls	To have a complete overview of the Fl0wer permissions.
tar cvf /tmp/supportinfo/fl0werconf.tar /opt/fl0wer/etc	To check the Fl0wer configuration.
tar cvf /tmp/supportinfo/fl0werlog.tar /opt/fl0wer/log	To check the latest Fl0wer error messages.
tar cvf /tmp/supportinfo/fl0werlist.tar /opt/fl0wer/iplist	To check the Fl0wer IP lists.
tar cvf /tmp/supportinfo/fl0werlua.tar /opt/fl0wer/luascripts	To check the Fl0wer LUA scripts.
uname -a	To have an overview of the system.
df -h	To understand available storage space.
fdisk -l	To understand available storage space.
pvdisplay	To understand available storage topology.
vgdisplay	To understand available storage topology.
lvdisplay	To understand available storage topology.
ifconfig -a	To understand the IP network configuration.
ip addr show	To understand the IP network configuration.
netstat -rn	To understand the network routing configuration.
ip route show	To understand the network routing configuration.
netstat -an	To have a deep understanding of what is happening at the network level.
netstat -su	To have a deep understanding of what is happening at the network level.
ss -a	To have a deep understanding of what is happening at the network level.

Command to be executed	**Reason**
dmidecode	To understand on which hardware is Fl0wer running.
dmesg	To have visibility about last kernel messages.
dpkg -l	To see the installed software versions.
rpm -qa	To see the installed software versions.

Once the /tmp/supportinfo.tar file is created, it should be sent as an attachment to the support request e-mail to support@fl0wer.me .

Support requests not providing the above said information attachment will be ignored and deleted.

Fl0wer Configuration File Example

The following section describes an example configuration file (*/opt/fl0wer/etc/fl0wer.conf*) that can be used for a normal-sized traffic collector and information-enriching system.

```
##################################
# Generic parameters
##################################
applog = /opt/fl0wer/log/fl0wer.log
errlog = /opt/fl0wer/log/errors.log
perflog = /opt/fl0wer/log/perf.log
luascriptdir = /opt/fl0wer/luascripts
enable_syslog = no
enable_perflog = no
maxthreads = 2
netflow_datafile = "/opt/fl0wer/data/netflowdata.flw"

fl0wer_ipaddress = "1.1.1.1"
track_fl0wer_dnsqueries = 0
silent_decoding = yes

###########################
# How the daemon should run
###########################
user = fl0wer
group = fl0wer

###########################
# Where is the license file
###########################
license_file = /opt/fl0wer/etc/license.lic

###########################
# Features to be used
###########################
enable_dns = yes
enable_nbar = no
enable_npar = yes
enable_nst = yes
enable_sentry = yes
enable_ip_cache = yes

enable_l2 = yes
enable_lua = yes
enable_listen_ipv6 = no
enable_relationship = yes

enable_torlist = yes
enable_dnslist = yes
enable_ntplist = yes
enable_bgplist = yes
enable_mynetlist = yes
enable_trafficrules = yes

ipv4_hostcache_limit = 100000
ipv6_hostcache_limit = 100000

enable_geoip       = no
dbgeo_ASNUM        = "/opt/fl0wer/geo/GeoIPASNum.dat"
dbgeo_ASNUM6       = "/opt/fl0wer/geo/GeoIPASNumv6.dat"
dbgeo_City         = "/opt/fl0wer/geo/GeoIPCity.dat"
dbgeo_ISP          = "/opt/fl0wer/geo/GeoIPISP.dat"
dbgeo_IPOrg        = "/opt/fl0wer/geo/GeoIPOrg.dat"
dbgeo_LiteCity     = "/opt/fl0wer/geo/GeoLiteCity.dat"
dbgeo_LiteCity6    = "/opt/fl0wer/geo/GeoLiteCityv6.dat"
dbgeo_Netspeed     = "/opt/fl0wer/geo/GeoIPNetSpeed.dat"
dbgeo_Country6     = "/opt/fl0wer/geo/GeoIPv6.dat"
geoip_fast         = yes

###########################
# Storage Format for both
# raw and flow files
###########################
storage_format = analytic

###########################
# Buffering
###########################
maxnetbuf=128
thread_nst_buffers = 100
buffers = 2000
flows_to_keep_in_ram = 2000
```

```
###########################
# Services to match
###########################
torlist  = "/opt/fl0wer/iplist/torlist.txt"
dns4list = "/opt/fl0wer/iplist/dns4list.txt"
dns6list = "/opt/fl0wer/iplist/dns6list.txt"
ntp4list = "/opt/fl0wer/iplist/ntp4list.txt"
ntp6list = "/opt/fl0wer/iplist/ntp6list.txt"
mynetlist = "/opt/fl0wer/iplist/mynetlist.txt"

###########################
# Network ports
###########################
netflow_port = 2056
clientport=7443

#######################################################
# The certificates for the TLSv12/SSLv2 JSON API
#######################################################
pem_cert_datafile = /opt/fl0wer/etc/certificate.pem
pem_key_datafile = /opt/fl0wer/etc/certificate.key
disable_tls = no
enable_dyncertfile = no

##########################################
# Example Definition of internal networks
##########################################
#network hurricane
#{
#        subnet = 2001:490:1f76:1c7::2
#        netmask = 64
#        description = "Tunnel Hurricane Electric"
#}

#network ip6test
#{
#        subnet = 2001:490:1b5b::
#        netmask = 48
#        description = "Internal IPv6 Testing"
#}


#network public
#{
#        subnet = 193.99.192.180
#        netmask = 30
#        description = PUBLICIP
#}

#network ubiquiti
#{
#        subnet = 10.100.14.0
#        netmask = 24
#        description = "Ubiquiti testing"
#}
#network mainisp
#{
#        subnet = 10.100.13.0
#        netmask = 24
#        description = "Main Internet subnet"
#}

#network backupisp
#{
#        subnet = 10.100.12.0
#        netmask = 24
#        description = "Backup Internet subnet"
#}
#
#network cisco
#{
#        subnet = 10.100.11.0
#        netmask = 24
#        description = "Router Internal"
#}
#
#network opnsense
#{
#        subnet = 10.100.10.0
#        netmask = 24
#        description = "Opnsense subnet"
#}
#

#network frontend
#{
#        subnet = 10.0.10.0
#        netmask = 24
#        description = Frontend
#}
```

```
#
#network backend
#{
#        subnet = 10.0.20.0
#        netmask = 24
#        description = Backend
#}

#network mgmt
#{
#         subnet = 10.0.30.0
#         netmask = 24
#         description = "Management Network"
#}
#
#network test
#{
#         subnet = 10.0.40.0
#         netmask = 24
#         description = "Testing Nework"
#}
#
#network transport
#{
#         subnet = 10.0.50.0
#         netmask = 24
#         description = Transport
#}
#network wifi
#{
#         subnet = 10.0.60.0
#         netmask = 24
#         description = "Wifi"
#}

#network SXT2
#{
#        subnet = 10.0.70.0
#        netmask = 24
#        description = "SXT2 Antenna"
#}


#network wifiguest
#{
#        subnet = 10.0.90.0
#        netmask = 24
#        description = "Wifi guest"
#}

##################################
# Traffic rules
##################################
traffic_rule mikrotik_discovery
{
        exporter = any
        ipversion = any
        protocol = any
        src_addr = any
        src_mask = 0
        src_port = 5678
        dst_addr = any
        dst_mask = 0
        dst_port = 5678
        maxpacketsize = any
        description = "Mikrotik neighbour discovery protocol"
        classification = normal
        store = no
}

#traffic_rule dns
#{
#         exporter = any
#         ipversion = any
#         protocol = any
#         src_addr = any
#         src_mask = 0
#         src_port = any
#         dst_addr = any
#         dst_mask = 0
#         dst_port = 53
#         maxpacketsize = any
#         description = "DNS Traffic"
#         classification = normal
#         store = yes
#}

#traffic_rule http
#{
#        exporter = any
#        ipversion = any
#        protocol = tcp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = 80
#        maxpacketsize = any
#        description = "HTTP Traffic"
#        classification = normal
#        store = yes
#}

#traffic_rule ntp_req
#{
#        exporter = any
#        ipversion = any
#        protocol = udp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = 123
#        maxpacketsize = any
#        description = "NTP Requests"
#        classification = normal
#        store = yes
#}

#traffic_rule ntp_reply
#{
#        exporter = any
#        ipversion = any
#        protocol = udp
#        src_addr = any
#        src_mask = 0
#        src_port = 123
#        dst_addr = any
#        dst_mask = 0
#        dst_port = any
#        maxpacketsize = any
#        description = "NTP Replies"
#        classification = normal
#        store = yes
#}

#traffic_rule https_dst_ipv6
#{
#        exporter = any
#        ipversion = 6
#        protocol = tcp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = 443
#        maxpacketsize = any
#        description = "IPv6 HTTPS"
#        classification = normal
#        store = yes
#}

#traffic_rule icmp_big
#{
#        exporter = any
#        ipversion = any
#        protocol = icmp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = any
#        maxpacketsize = 64
#        description = "ICMP Traffic bigger than 64 bytes"
#        classification = suspicious
#       store = yes
#}

traffic_rule igmp_big
{
        exporter = any
        ipversion = 4
        protocol = igmp
        src_addr = any
        src_mask = 0
        src_port = any
        dst_addr = any
        dst_mask = 0
        dst_port = any
        maxpacketsize = 256
        description = "IGMP Traffic bigger than 256 bytes"
        classification = suspicious
        store = yes
}

#traffic_rule HSRP
#{
#        exporter = any
#        ipversion = 4
#        protocol = udp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = 1985
#        maxpacketsize = any
#        description = "HSRP Traffic"
#        classification = normal
#        store = yes
#}

#traffic_rule VRRP
#{
#        exporter = any
#        ipversion = 4
#        protocol = vrrp
#        src_addr = any
#        src_mask = 0
#        src_port = any
#        dst_addr = any
#        dst_mask = 0
#        dst_port = any
#        maxpacketsize = any
#        description = "VRRP Traffic"
#        classification = suspicious
#        store = yes
#}

traffic_rule finger
{
        exporter = any
        ipversion = any
        protocol = tcp
        src_addr = any
        src_mask = 0
        src_port = any
        dst_addr = any
        dst_mask = 0
        dst_port = 79
        maxpacketsize = any
        description = "Finger Traffic"
        classification = suspicious
        store = yes
        action = syslog
}

traffic_rule telnet
{
        exporter = any
        ipversion = any
        protocol = tcp
        src_addr = any
        src_mask = 0
        src_port = any
        dst_addr = any
        dst_mask = 0
        dst_port = 23
        maxpacketsize = any
        description = "Telnet Traffic"
        classification = suspicious
        store = yes
        action = syslog
}

traffic_rule rsh
{
        exporter = any
        ipversion = any
        protocol = tcp
        src_addr = any
        src_mask = 0
        src_port = any
        dst_addr = any
        dst_mask = 0
        dst_port = 513
        maxpacketsize = any
        description = "Remote Shell Traffic"
        classification = suspicious
        store = yes
        action = syslog
}
```

The Performance Test Suite

During development of the Fl0wer server, I wanted to be sure to provide enterprise class performance to the users. While the market-share that Fl0wer targets should not necessarily have very intensive Netflow data streams, I wanted to give freedom to users to use it the way they want.

The model used to test how much incoming flows Fl0wer can handle is based on heavy stressing using the tcpreplay tool (http://tcpreplay.synfin.net). I used the version 4.10 building it by myself.

Netflow packets are simple UDP packets sent by collectors that does not expect an answer, so the tcpreplay method works pretty well. The key thing taken in account is that, while there is no guarantee that an UDP packet reaches its destination (it's a part of the name, Unreliable Data Protocol!), we should not see packet loss due to incapacity of the server to process all incoming data streams.

Fl0wer, in its licensed version, is a powerful multithreaded server and the distribution of packets between threads is handled by the Linux/UNIX kernel by means of using the SO_REUSEPORT system call. This means that each flow processing thread can receive packets to process and the load balancing of packet distribution is done by the kernel.

The stress test script takes a set of pre-built Netflow V5, Netflow V9 and IPFIX packets from about 50 different sources, adjusts them to be sent to the user specified IP and MAC address and throws them to the Fl0wer server at the user set rate (in terms of Megabit per second). On Linux, packet loss can be monitored real-time using the *netstat -su* command, and UDP queues can be monitored in real-time using the command *ss -uamp | grep fl0wer* . On properly tuned, *relatively* high-end old hardware (dual Intel X5670 CPU with 24Gb of RAM), fl0wer has been measured to perform at over 200000 Flows per Second doing only flow collection and NPAR. Obviously, the more features you enable, the lower the Flow per Second will be.

The script */opt/fl0wer/perfsuite/testsuite.sh* script and the relative datafiles can be used, after proper configuration for your environment, to take measurements. Good Linux/UNIX system & networking skills are required to make it work, you really need an expert. No free support is provided in adjusting the script or tuning your server. The script and the Performance measurement scenario requires a Linux "exporter" system that sends the stress data.

►It is completely pointless to make a stress test on an evaluation version since it is limited to only one flow processing thread.

The testing iter works in this way:

1. Identify the Linux "exporter" system that will throw traffic to the Fl0wer server. It is strongly suggested to have it on the same IP subnet of the Fl0wer server, since a router or a firewall could drop packets in the middle.
2. Copy the */opt/fl0wer/perfsuite* directory onto your Linux exporter system and work from there.
3. Edit and adjust the values in the build scenario script in *perfsuite/buildscenario.sh.* Note the TARGET (the Fl0wer server) IP and MAC address.
4. Configure properly the Fl0wer server (buffers, maxnetbuf, enabled features, kernel stack, etc.)

5. Start throwing the nf*_multi_target.pcap to the Fl0wer server, starting first with slow throughput (5 Mbit/s is a good start) and using *htop*, *netstat -su* and *ss -uamp* check for packet loss, CPU usage, queues in the kernel and disk usage on the Fl0wer server. Then adjust the parameters and retry.

External Software Licenses

The Fl0wer server software makes use of some open-source or free external projects.

The projects are:

- OpenSSL
- libConfuse
- MaxMind GeoIP Legacy API
- libcurl
- sqlite3

All their respective licenses and web links are here reported.

OpenSSL and SSLeay License

The Fl0wer server makes use of the OpenSSL software for encryption and other functions. The software is available at http://www.openssl.org/ . The licensing conditions at the date of writing (20 March 2017) this manual are available at: https://www.openssl.org/source/license.html and are hereby verbatim reported.

libConfuse license

The Fl0wer server makes use of the libConfuse software for handling of configuration files. The software is available at https://github.com/martinh/libconfuse . The licensing conditions at the date of writing (20 March 2017) this manual are subject to the ISC License, a permissive license that lets people do anything with the code with proper attribution and without warranty. The full text license is available at: https://github.com/martinh/libconfuse/blob/master/LICENSE and is hereby verbatim reported.

MaxMind GeoIP Legacy API License

The Fl0wer server makes use of the MaxMind GeoIP Legacy API software for handling the MaxMind databases for IP address georeferencing. The software is available at https://github.com/maxmind/geoip-api-c . The licensing conditions at the date of writing (20 March 2017) this manual are subject to the GNU LGPL License. The Fl0wer server simply uses the library without modifying it, so it falls under the category “work that uses the Library”. The paragraph 5 of the GNU LGPL clearly states that this kind of work falls outside the scope of the GNU LGPL.

“A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.”

The full text license is available at: https://github.com/maxmind/geoip-api-c/blob/master/LICENSE and is hereby verbatim reported. The Fl0wer software does not distribute the GeoLite Country database.

libcurl License

The Fl0wer server makes use of the libcurl software for handling of some functions. The software is available at https://curl.haxx.se/libcurl/ .

The full text license is available at: https://curl.haxx.se/docs/copyright.html and is hereby verbatim reported.

Curl and libcurl are true Open Source/Free Software and meet all definitions as such. It means that you are free to modify and redistribute all contents of the curl distributed archives. You may also freely use curl and libcurl in your commercial projects.

Curl and libcurl are licensed under a MIT/X derivate license, see below.

There are other computer-related projects using the name curl as well. For details, check out our position on the curl name issue.

The curl license

Sqlite3 License

The Fl0wer server makes use of the sqlite3 software for handling of user management and other functions. The software is available at https://sqlite.org/index.html .

The full text license is available at: https://sqlite.org/copyright.html and is hereby verbatim reported.

SQLite Is Public Domain

All of the code and documentation in SQLite has been dedicated to the public domain by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed affidavits are stored in a firesafe at the main offices of Hwaci. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

The previous paragraph applies to the deliverable code and documentation in SQLite - those parts of the SQLite library that you actually bundle and ship with a larger application. Some scripts used as part of the build process (for example the "configure" scripts generated by autoconf) might fall under other open-source licenses. Nothing from these build scripts ever reaches the final deliverable SQLite library, however, and so the licenses associated with those scripts should not be a factor in assessing your rights to copy and use the SQLite library.

All of the deliverable code in SQLite has been written from scratch. No code has been taken from other projects or from the open internet. Every line of code can be traced back to its original author, and all of those authors have public domain dedications on file. So the SQLite code base is clean and is uncontaminated with licensed code from other projects.

LUA Interpreter

The Fl0wer server makes use of the LUA 5.3 embeddable interpreter to execute user-defined scripts on receipt of flows and templates. The software is available at https://www.lua.org .

The full text license is available at: https://www.lua.org/license.html and is hereby verbatim reported.

UT Hash

The Fl0wer server makes use of the UT Hash macros to handle hash-tables for IPCache and other data structures. The software is available at http://troydhanson.github.io/uthash/index.html .

The full text license is available at: http://troydhanson.github.io/uthash/license.html and is hereby verbatim reported.