Purpose

This document gives an overview of how the backend of Nagios Network Analyzer functions, and general tips. This documentation will give you a better idea of how Network Analyzer works, where files are located, how they are updated and some basic troubleshooting advice.

Target Audience

This document is intended for use by network administrators and technicians who want to better understand how Nagios Network Analyzer works.

Data Location

All of the data that Nagios Network Analyzer collects is kept in the `/usr/local/nagiosna/var` directory. There is no other directory that information is written to in regards to actual flow data. Under this `var/` directory, there will exist sub-folders that correspond to individual Sources. The Source names get sanitized to make them filesystem safe, they will show under `var/` in a form that is much like their given name.

For instance, a Source called `This Has Spaces` gets changed to `ThisHasSpaces` and a new directory in `var/` is created. To access the flow information for `This Has Spaces`, you would access `var/ThisHasSpaces`. Inside this directory, there are three items of interest:

- `flows/` directory
  - Contains the binary files that `nfcapd` produces. These are used to get granular Netflow information using `nfdump`.
- `bandwidth.rrd`
  - The `bandwidth.rrd` is used to generate on-demand traffic graphs.
- `views/` directory
  - The views/ directory is created when a View is associated with this particular Source.

How Data Is Captured

Nagios Network Analyzer relies on `nfcapd/sfcapd` to capture flow data. Each individual Source has a process that is spawned for it to capture data. Each process opens up a port and listens on this port. Any valid Netflow data it receives will be processed and recorded in the `flows/nfcapd.current` file and will be expired into a new file, `flows/nfcapd.<datestamp>`, whenever the system clock hits a 5 minute interval.

Once this new file has been made, a callback script gets kicked off by the `nfcapd` process. This callback script, `reap_files.py`, is in charge of a couple things:

- Updating Views associated with the process's source
- Updating the `bandwidth.rrd` for the source
- Running Nagios checks on the source

This callback script is run every 5 minutes, or every time a new nfcapd file is made. I cannot stress the importance of this concept enough when troubleshooting or adding functionality to Nagios Network Analyzer. This callback script logs to `nagiosna/var/backend.log`.

The bottom line: If there is information updated, or some action that happens based on the arrival of new Netflow data, it happens somewhere in the callback script.

Also on the creation of a new nfcap file, `nfexpire` is also called. `nfexpire` is independent of the callback script, and will remove nfcap files that are older than the data lifetime that is specified at source creation.

How Bandwidth RRDs Are Updated

The bandwidth RRDs hold how much traffic has been observed at each update interval. They hold information about Bytes, Flows, Packets, Bytes/Sec, Packets/Sec and Bytes/Packet. These RRDs are updated at each 5 minute interval and are numbers that are derived from running nfdump on all newly created nfcap files. The callback script `reap_files.py` is how the RRD files are are updated.

How Views Are Handled

Views are useful for Nagios Network Analyzer as they keep the space used down, and allow the user to have as much granularity they wish for particular aspect of their network traffic. As mentioned in the previous section, Views are handled and maintained by the callback script.

The train of logic that is followed when updating views in the callback goes like this:

- Does the `view` folder and RRD exist? If not, make them.
- Run the View limiter on the newly created nfcapd file, pipe the results to an identically named file in the `view/<viewname>` folder.
  - This file created by the piping, contains only information that pertains to the View.
- Update the View bandwidth RRD. Note that the View bandwidth RRD is not called `bandwidth.rrd`, but is instead called `views/<view name>.rrd`.

This is how Views are created and updated by Nagios Network Analyzer.

How Checks/Notifications Are Handled

Checks are processed in the end of the callback script. The callback script itself calls the `netflow_checks.py` script which runs the actual checks by running nfdump on the latest nfcap file. Once it gets the numbers from the nfcap file, it runs the `notify.py` script, which queries the SQL database for who should be notified and how they should be notified.

Email is handled by the built-in Python mail server, so as of now, any configuration for email or using a mail server would have to be edited into the send_email function in `notify.py`. However, please consider using Nagios to handle emails, as the `notify.py` also sends out NRDP notifications to Nagios, and the email functionality built into Nagios Network Analyzer was meant to provide a solution, but does not support the robust configuration for notifications that Nagios Core or Nagios XI does.

Also of note is the SNMP trap sending which uses the net-snmp binary to send SNMP traps using the bundled MIB (located in `trunk/mib`).

General Troubleshooting Advice

The heartbeat of Nagios Network Analyzer is the `nfcapd` process. Keep in mind, each Source gets its own `nfcapd`/`sfcapd` process. So when troubleshooting issues arise with the backend, its generally best to make sure that the Source-in-question's nfcapd is running. This is done by logging onto the Network Analyzer server as the root user and grepping through the ps output from the command line:

```
ps aux | grep <directory of source>
```

If this isn't running, then the problem begins there, and can be remedies by restarting the process via:

```
/usr/local/nagiosna/bin/nagiosna restart
```

This will restart each process that Nagios Network Analyzer is aware of, however you can be more selective by typing:

```
/usr/local/nagiosna/bin/nagiosna <Filesystem Safe Name of Source> restart
```

Note: You will reference the name of the **directory** for the Source in question and not the name of the Source in Network Analyzer.

Once you have verified that nfcapd is indeed running for the Source in question, its time to check to make sure that there is nothing weird showing up in the `var/backend.log`. Any known failures that are occurring that can cause a host of issues will show up there. If the callback script ever has issues, it will be logged here with the reason.

Another major technique that will be very helpful, is determining whether or not traffic is being received by the Nagios Network Analyzer server on a specific port. This is very easily done via the use of `tcpdump`. This will absolutely tell you whether or not Netflow traffic is coming in. Simply run the command:

```
tcpdump dst port <port that traffic is supposed to be coming in on>
```

If there is no traffic reported here, you'll need to go back to the device that is allegedly sending Netflow data. If you see the traffic reported then you may also want to check that the local firewall has the correct rules created. These are created automatically by Network Analyzer but can be checked by running:

```
iptables --list
```

Some nfdump commands are helpful for troubleshooting incoming Netflow traffic. To ensure that data is actually coming from the Netflow device and is being captured by the nfcap process, navigate to the `flows/` directory of a Source in question and find the newest nfcap file that is not `nfcap.current` and run:

```
nfdump -r <newest nfcap file>
```

If this shows any data at all, it should also be present in the web interface, if this is showing zeros, then the web interface should be showing zeros as well.

Finishing Up

This completes the documentation on understanding the Nagios Network Analyzer backend.
If you have additional questions or other support related questions, please visit us at our Nagios Support Forums:

https://support.nagios.com/forum

The Nagios Support Knowledgebase is also a great support resource:

https://support.nagios.com/kb