Netflow Program Logic Flow

The netflow accounting system based on JKFlow is a dynamic and XML-configurable reporting tool for network traffic. This page describes a logic flow of the program, including the behaviour of the code contained in flowscan and JKFlow.pm as well as how the configuration in JKFlow.xml and flowscan.cf affects this behaviour.

Flowscan

This is the main executable file. All other modules and dependent sub-routines are called from within this file. Flowscan assumes that flow-files containing raw flow information are being constantly generated in a folder on the system. The location of this folder is specified in the flowscan.cf by the identifier FlowFileGlob. Generally flow-files are named so that their names indicate a timestamp for when that file was generated. In the current configuration flow-files are being generated every minute. For example the following listing of /var/flows/flows shows two flow-files currently in the directory.

Sample listing of a flow-file directory

```
akbar@iepm-resp $ ls -rtl /var/flows/flows
total 552
-rw-r--r--    1 akbar    sg         224688 Apr 11 19:06 USA-ft-v05.2007-03-21.121300-0400
-rw-r--r--    1 akbar    sg         261616 Apr 11 19:06 USA-ft-v05.2007-03-21.121200-0400
```

The basic job of flowscan is to take up the files present in the flow-file folder (such as /var/flows/flows) and process them using a reporting module such as JKFlow.pm in order of their timestamps. It runs in an infinite loop and keeps checking /var/flows/flows for new flow-files and if there are any present it processes them and deletes each flow-file as it is processed by the reporting module. The choice of reporting module is also configurable and can be specified in flowscan.cf using the identifier ReportClasses.

As shown in the code below, flowscan loads the names of the classes into an array and then later "includes" them by doing an eval on all the classes in the array. Usually only one reporting module such as JKFlow.pm is used.

```
# Set the default options from the configuration file:
$c = new ConfigReader::DirectiveStyle;
$c->directive('Verbose');
$c->directive('WaitSeconds');
$c->required('FlowFileGlob');
$c->required('ReportClasses');

# terapaths monitoring
$c->required('SyslogFacility');

$c->load("${FindBin::Bin}/${FindBin::Script}.cf");
$flowfileglob = $c->value('FlowFileGlob');
$opt_w        = $c->value('WaitSeconds');
$opt_v        = $c->value('Verbose');
@classes      = split( m/\s*,\s*/, $c->value('ReportClasses') ); # loads all the class names given in flowscan.
cf into an array

|   |
|   |
|   |

foreach my $class (@classes) { # includes each of the classes
    eval "use $class";
    die "$@" if $@;
}
```

The main processing in flowscan takes place in an infinite while loop which keeps on processing flow-files as they are generated in the folder /var/flows /flows. The main steps that this while loop takes are:

1. For all the flow-files currently in /var/flows/flows it sorts them by their names (which represents time-stamps).
2. Next it takes each file in the sorted list and
    a. Generates a new object for each of the reporting classes for this flow-file's processing. Usually since only JKFlow.pm is the only reporting class only one object is instantiated.
    b. Calls the Cflow::find function passing the references of the "wanted" function in flowscan alongwith the pathname of the flow-file currently being processed. Cflow:find calls the wanted sub-routine for each flow-record present in the flow-file passed. The subroutine wanted is defined in flowscan but it is merely a stub which calls the wanted subroutines of all the reporting classes in the @classes array(i.e. JKFlow.pm in this case). The wanted sub-routines defined in reporting classes (such as JKFlow.pm) are basically designed so that they are called once for each individual flow-record. In case of JKFlow.pm as the wanted sub-routine is called for each flow-record, it updates the values of the data structures containing the network flow accounting and scoreboarding information which are maintained by JKFlow. A more detailed description of the wanted function is given in the description of JKFlow

c. Calls the report function in flowscan. Once again the report function is also a stub which inturn calls report functions in all the reporting classes. The report function in case of JKFlow.pm handles the maintainence of scoreboards representing the topN information as well as writing out RRD reports of the data based on information gathered until now.
d. Finally destroys the array of reporting classes and objects. *A key point to note here is that although the objects are destroyed the global data structures for flow-accounting remain intact so that when the objects are created for the processing of the next flow-file, they access the same data strcutures as for the previous flow-file.*