Description

The suggestion is to be able to output the Suricata statistics (stats.log) in a format that is easily processed by other applications. This would make the viewing and use of the statistics easier and faster for the users. There could also be added a way to indicate what format and or level of detail that is wanted (ex: aggregating thread values at "low", write all possible data at "high")

Today:

- stats:
  - enabled: yes
  - filename: stats.log
  - interval: 8
  - append: yes/no

Possible future:

- stats:
  - enabled: yes
  - format: plain/json/bson/csv/...
  - details: low/medium/high
  - interval: 10
  - append: yes/no

History

#1 - 06/29/2014 01:37 PM - Peter Manev

Similar to
https://redmine.openinfosecfoundation.org/issues/1036

#2 - 10/21/2014 03:56 PM - Andreas Moe

Started working on a branch for this. Just a simple stats.format key in the yaml file (as shown above), and using the current data available in the counters.c file. Using libjansson for the JSON creation. Any suggestions to JSON-Structure? I was thinking something like:

```json
[
  "metadata": {
    "date": "21/10/2014",
    "uptime": {
      "days": 1,
      "hours": 2,
      "minutes": 3,
      "seconds": 4,
    },
  },
  "Counters": {
    "TM Name": {
      "counter": Value,
      "counter": Value,
      "counter": Value,
    },
    "TM Name": {
  }
]```
Great that you want to have a look Andreas. However, quite a bit of code already exists here: https://github.com/inliniac/suricata/pull/1010

Ahh, didn't see that pull request. Looks like a lot has been done. Is the wish for a Stats logger API (as you Victor commented on the pull request) what is stopping it from being merged or are there any other missing features / bugs?

The stats logger api doesn't exist yet, so this will have to be created first.

Andreas. I'd be happy to work with you to finish this. I was hoping that Victor would propose a stats logger api. I ran with what was there. Perhaps Victor can propose something and Andreas can help me implement it on top of what has been done already.

I'm not happy with the structure though. I've tried both Kibana and Graphite (fed by Logstash), but neither can easily take it in. One of the issues is that the "TM Name" is different for each capture method, capture config and runmode. So having a dashboard that works for all is going to be impossible, while this is normally the strength of our eve output.

When feeding Logstash to Graphite I found that it didn't really like the nesting (e.g. tcp:{syn:10} for counter tcp.syn). But when using the 'fields_are_metrics' option it didn't like the wrapping of it all in 'stats' either.

Minor disappointment was that Kibana 3 couldn't display multiple data series in the histogram view (e.g. decoder.pkts and decoder.invalid), although this is likely to be added in Kibana 4.

So I think what we need is either different format that 'just works', or we should supply decent example configs for making good use of this in Kibana and/or Graphite.

Feature #490 looks at the issue of non-aggregated stats. Could (with the implementation you have given Victor) this be included now, by say a flag in the YAML (eg. aggregated yes/no) and sum up the similar counter values?

https://github.com/inliniac/suricata/pull/1508