NAME

sc_minrtt — Manage RTT input to sc_hoiho

SYNOPSIS

sc_minrtt [**-c**] [**-d** *db-file*]

sc_minrtt [**-i**] [**-d** *db-file*] [**-R** *regex*] *file ...*

sc_minrtt [**-p** *process-mode*] [**-d** *db-file*] [**-r** *router-file*] [**-t** *threadc*] [**-V** *vploc-file*]

DESCRIPTION

The **sc_minrtt** utility processes RTT data collected with sc_pinger(1) using a series of vantage points into a format for sc_hoiho(1) to use. **sc_minrtt** builds an sqlite3 database, which it uses to store RTT samples for faster processing. The intended workflow using **sc_minrtt** is to first create a blank sqlite3 database using **-c**, import the RTT samples from warts(5) files using **-i**, and then use the database to create the RTT constraints file for sc_hoiho(1) using **-p.** The supported options to **sc_minrtt** are as follows:

-c specifies that a sqlite3 database file is to be created.

-d *db-file*
specifies the name of the sqlite3 database file to use.

-i specifies that the RTT samples in the supplied warts(5) files collected with sc_pinger(1) should be imported into the sqlite3 database.

-p *process-mode*
specifies the processing to do on the RTT samples, either mode 1 or 2. Mode 1 identifies VPs that might be contributing spurious RTT samples. Mode 2 dumps the minimum set of RTT constraints from the collected RTT samples.

-r *router-file*
specifies the name of a corresponding router file that maps IP addresses to routers. The format of this file is the same as that supplied to sc_hoiho(1).

-R *regex*
specifies the regular expression to apply to input file names that extracts a vantage point name. The extracted vantage point name must subsequently match an entry in the vploc file supplied with the **-V** option when processing the samples.

-t *threadc*
specifies the number of threads to use when processing the database.

-V *vploc-file*
specifies a file containing a mapping of vantage point names to lat / long coordinates.

EXAMPLES

Given a set of warts files named hlz2-nz.pinger.warts, ams7-nl.pinger.warts, cld-us.pinger.warts, and a blank database created with

sc_minrtt -c -d minrtt.sqlite

the following will import the RTT samples:

sc_minrtt -i -d minrtt.sqlite -R "([a-z]{3}\d*-[a-z]{2})\.pinger\.warts$" /path/to/*.pinger.warts

To dump the minimum set of RTT values providing constraints per IP address, use:

sc_minrtt -p 2 -d minrtt.sqlite -v vploc.txt

To dump the minimum set of RTT values per router for use by sc_hoiho(1), use:

sc_minrtt -p 2 -d minrtt.sqlite -v vploc.txt -r routers.txt

NOTES

sc_minrtt records which files it has imported in the database, so that it does not re-import the same file multiple times. It does not store the full path to the file, so all filenames need to be unique, even if they are stored in different directories.

sc_minrtt organizes entries in a binary blob for each IP address in the sqlite database. It is best to write the database to a file located on a tmpfs(5) filesystem, and then copy it to disk once it is created.

sc_minrtt attempts to determine responses that are forged by a middle-box close to the vantage point by looking for reply-TTL run-lengths involving many unique destinations. Internally, the threshold is a run of 50 unique destinations with the same reply-TTL run-length. This feature cannot currently be disabled or changed at runtime.

The format of the vploc.txt file can be one of two supported formats, either

```
ams7-nl 52.35 4.82
hlz2-nz -37.78 175.17
cld-us 32.88 -117.24
```

or

```
vp ams7-nl 52.35 4.82
vp hlz2-nz -37.78 175.17
vp cld-us 32.88 -117.24
```

The latter format has the string "vp" at the start of each line, and is the same format used by sc_hoiho(1), so you can supply the same vploc.txt file to both sc_hoiho(1) and **sc_minrtt**.

SEE ALSO

sc_hoiho(1), sc_pinger(1), sqlite3(1)

AUTHORS

sc_minrtt was written by Matthew Luckie. Shivani Hariprasad and Harsh Gondaliya developed code to emit the minimum set of RTT constraints per address or router.