NAME

sc_uptime — scamper driver to infer reboot windows for systems with IPv6 addresses.

SYNOPSIS

sc_uptime [**-a** *addressfile*] [**-d** *dbfile*] [**-l** *logfile*] [**-o** *outfile*] [**-O** *option*] [**-p** *port*] [**-S** *srcaddr*] [**-U** *unix-socket*]

sc_uptime [**-i**] [**-d** *dbfile*] [**-O** *option*] [*file ...*]

sc_uptime [**-r**] [**-d** *dbfile*] [**-O** *option*] [*ip ...*]

DESCRIPTION

The **sc_uptime** utility provides the ability to connect to a running scamper(1) instance and use it to collect data for inferring if and when a system with an IPv6 address rebooted. **sc_uptime** induces each system to send fragmented ICMP echo replies, with the goal of obtaining an incrementing Identifier (ID) field in the fragmentation header. If the system assigns ID values from a counter, and the counter resets to zero, the system rebooted in the interval between probes. Note, the technique does not work if the system does not send ICMP6 fragments, if fragments are blocked, or if the system assigns ID values randomly. **sc_uptime** implements a scalable algorithm to avoid probing systems that assign ID values at random or are unresponsive. Further information about the algorithm is found in the "see also" section. The supported options to **sc_uptime** are as follows:

-a *addressfile*
: specifies the name of the input file which consists of a sequence of IPv6 addresses to probe for reboots, one address per line. Any new addresses are added to the sqlite3 database for subsequent processing. Any addresses in the sqlite3 database but not in the addressfile continue to be probed.

-d *dbfile*
: specifies the name of the output sqlite3 database file to use

-i
: specifies that the IPID samples in the supplied warts(5) files should be imported into the sqlite3 database.

-l *logfile*
: specifies the name of a file to log progress output from **sc_uptime** generated at run time.

-o *outfile*
: specifies the name of the output file to be written. The output file will use the warts(5) format.

-O *options*
: allows the behavior of **sc_uptime** to be further tailored. The current choices for this option are:
  - **create-db:** create and initialise a brand new sqlite3 database. Use this option the first time **sc_uptime** is run and a state database or a data database has not been created.
  - **safe-db:** when importing into the data database, use safe operations. By default, **sc_uptime** enables optimizations that will speed up bulk data importing, at the expense of possible database corruption in the event power is lost during the import.
  - **vacuum-db:** perform a database vacuum before doing any operations.

-p *port*
: specifies the port on the local host where scamper(1) is accepting control socket connections.

-r
: specifies that **sc_uptime** should use the IPID samples in the sqlite3 database to infer reboots.

-S *srcaddr*
: specifies the unicast IPv6 source address to use in probes. This option is recommended when the prober has multiple or changing IPv6 addresses, as some probed systems maintain an IPID counter related to the source IPv6 address of the prober. If the source address of probes changes, the

returned IPID sequence could have a discontinuity unrelated to a reboot of the probed system.

-U *unix-socket*
specifies the name of a unix domain socket where scamper(1) is accepting control socket connections.

EXAMPLES

Given a set of IPv6 addresses contained in a file named addressfile.txt and a scamper process listening on port 31337 configured to probe at 30 packets per second started as follows:

```
scamper -P 31337 -p 30
```

the following command will create a sqlite3 database named state.sqlite to record probe state within, probe the addresses contained in addrs.txt once to collect raw data that can be used to infer reboot windows, and store the raw data in out1.warts:

```
sc_uptime -p 31337 -O create-db -d state.sqlite -a addrs.txt -o
out1.warts
```

To infer reboot windows, **sc_uptime** must be run in rounds. The next time **sc_uptime** is run, use:

```
sc_uptime -p 31337 -O create-db -d state.sqlite -o out2.warts
```

Import the IPID samples from out1.warts and out2.warts into a separate probe database:

```
sc_uptime -i -O create-db -d data.sqlite out1.warts out2.warts
```

Infer reboots across all imported data:

```
sc_uptime -r -d data.sqlite
```

To infer reboots just for IPv6 address 2001:DB8::1, use:

```
sc_uptime -r -d data.sqlite 2001:DB8::1
```

HINTS

Store the state.sqlite file on a memory file system to improve speed and reduce disk wear. In between invocations of **sc_uptime**, copy the state.sqlite file to disk. You can store probe data in the same database used to maintain state, but you should use a separate database. To obtain better performance, import the data into a time-series database.

NOTES

sc_uptime will only probe addresses in 2000::/3.

SEE ALSO

scamper(1), sc_ally(1), sc_ipiddump(1), sc_speedtrap(1), sc_wartsdump(1), sc_warts2text(1), sc_warts2json(1), sqlite(1), warts(5),

M. Luckie and R. Beverly, *The Impact of Router Outages on the AS-level Internet*, Proc. ACM/SIGCOMM Conference 2017.

R. Beverly, M. Luckie, L. Mosley, and k claffy, *Measuring and Characterizing IPv6 Router Availability*, Proc. PAM 2015.

AUTHORS

sc_uptime was written by Matthew Luckie and Robert Beverly.