Flamethrower

Jan Včelák
May 13 2019, DNS-OARC 30 (Bangkōk)
Flamethrower: DNS Performance Testing Tool

https://github.com/DNS-OARC/flamethrower
Apache Public License 2.0

Built as an alternative to \textit{dnsperf}.

Initial requirements:
\begin{itemize}
\item Better source port distribution
\item Solid TCP support
\item Realistic query rate patterns
\item Possibility to integrate into a CI/CD pipeline
\end{itemize}
Flamethrower
Quick Start

> flame \
- r foo.example.test.
- T SOA \n- Q 2000 \n- l 5 \n a.ns.example.test

flaming target "a.ns.example.test" (192.0.2.100) on port 53 with 10 concurrent generators, each sending 10 queries every 1ms on protocol udp
query generator [static] contains 1 record(s)

----------------------------------------------------
----------------------------------------------------

runtime : 7.88192 s
total sent : 11756
total rcvd : 9223
min resp : 23.3888 ms
avg resp : 27.9633 ms
max resp : 33.3626 ms
avg rps : 1834
avg qps : 2350
avg pkt : 31.92 bytes
timeouts : 2533 (21.5464%)
responses :
SERVFAIL: 441
NOERROR: 8782
Flamethrower Internals

Flamethrower is written in C++14 (17).

Notable dependencies:

- LDNS - DNS library
  https://nlnetlabs.nl/projects/ldns/about/
- uvw - asynchronous I/O (modern C++ wrapper for libuv)
  https://github.com/skypjack/uvw
Flamethrower Architecture

payload generator → traffic generator → rate control → metrics
Payload Generators

- **static** — Single query name and type.
- **file** — Consumes dnsperf-compatible input file (i.e. pairs of query name and type).
- **randompkt** — Random binary garbage.
- **numberqname** — Names prefixed with a random numeric label:
  - 1234.example.test.
  - 42.example.test.
  - ...
- **randomqname** — Names prefixed with random binary labels:
  - j4kJx\000zz3d\064sa.example.test.
  - bo4._kf\042.example.test.
  - ...
- **randomlabel** — Names prefixed with random non-binary labels:
  - XRY4HFY9dpItTb.example.test.
  - ady1.cIo.ZUq\V4gji.example.test.
  - ...
Traffic Generator

Flamethrower runs in a **single thread** only.
Max query rate ~100k queries/s on a single core.

Traffic generator config options:

- `-c` Number of concurrent traffic generators.
- `-q` Number of queries to send in a batch.
- `-d` Time (delay) between batches.

Protocol selection option:

- `-P udp` Default. One bound UDP socket per traffic generator.
- `-P tcp` One TCP sessions per traffic generator. Keeps the session open.
- `-P dot` DNS-over-TLS.
Rate Control

Constant rate:  `flame -Q 1000`
Variable rate:  `flame --qps-flow '1000,60000;100,60000;...'`
Metrics


run id : c50c74a39823a4c1
run start : 2019-02-01T22:38:01Z
runtime : 122.894 s
total sent : 120887
total rcvd : 120581
min resp : 62.7252 ms
avg resp : 41.2032 ms
max resp : 94.3204 ms
avg rps : 987
avg qps : 998
avg pkt : 23.4042 bytes
tcp conn. : 0
timeouts : 306 (0.253129%)
bad recv : 0
net errors : 0
responses :
  NOERROR: 120581

Machine friendly variant available with:

> flame -o results.json

Samples

10 queries every 500 ms on 100 source ports over UDP (= 2000 q/s). Queries read from a dnsperf-compatible input file.

```
> flame -q 10 -d 500 -c 100 \
   -g file -f queries.txt \
   ns.example.test
```

10 queries every 250 ms on 250 source ports over UDP (= 10k q/s) with a query rate decay. Queries for a random subdomains on example.test. Metrics written into a JSON file. Target servers loaded from a file.

```
> flame -q 10 -d 250 -c 250 \
   --qps-flow '10000,5000;5000,5000;2500,5000;1250,5000;1000,40000' \
   -g randomlabel lblsize=10 lblcount=1 count=140000 -r example.test -T AAAA \
   -o peak.json \
   --targets servers.txt
```

1 “query” every 100 ms on 100 concurrent IPv6 TCP sessions (= 1000 q/s). Random “garbage” up to 100 bytes.

```
> flame -q 1 -d 100 -c 100 \
   -g randompkt count=10000 size=100 \
   -F inet6 -P tcp ns.example.test
```
New Features

Recently finished

‣ DNS-over-TLS support
‣ Target multiple servers

In progress

‣ DNS-over-QUIC support

Wanted

‣ Multi-processor support
‣ Query rate pattern from a pcap
‣ Improve performance
‣ Source address spoofing
‣ DNS-over-HTTP support
Thank you!

https://github.com/DNS-OARC/flamethrower