Comparison of RRL in BIND, Knot DNS and NSD

DNS OARC Spring Workshop
Dublin, May 2013
Dave Knight
Background

• In March an L.root-servers.net node in Hamburg was being used in an amplification attack

• Mitigated with NSD RRL

• Felt that the decrease in outbound traffic was smaller than we expected

• Decided to do some comparison testing of the different RRL implementations
Comparison

- Compared RRL performance in the following implementations
  - BIND9  (9.9.2)
  - BIND10  (20130503)
  - Knot DNS  (1.2)
  - NSD3  (3.2.15)
  - NSD4  (4.0.0b4)
Lab

• Used the OARC lab for this work
  ▶ Uncontentious place for others to bring their data
  ▶ Others can easily be given access to what I did to run it for themselves
  ▶ Many thanks to Keith, William and Geoff for making this work
Lab environment

• 3 servers
  ‣ Query generator
  ‣ Nameserver
  ‣ Response collector
• Running Ubuntu 12.04
• GigE switch
Queries

- 25 minutes of traffic captured at ham01
- 5 x 5 minute pcaps
- Stripped out TCP
- Replayed toward the nameserver with tcpreplay
- Static route to L-root pointed at the nameserver
Nameservers

- Installed Non-RRL and RRL builds of all nameservers (except Knot, in 1.2 it’s built in by default so I used the Ubuntu package)

- Ran each nameserver with Non-RRL and RRL configurations. Config files were kept as simple as possible. Didn’t optimize for performance, only care about RRL.

- Configured with L-root service addresses
Response Collector

• Nameserver default route pointed at the collector

• Collected responses with tcpdump
Tests

• BIND9 and BIND10 use the Redbarn spec
• Knot and NSD don’t
• Comparing nameservers configured with
  ‣ No RRL
  ‣ RRL enabled with that implementations defaults
  ‣ RRL enabled with the Redbarn defaults
  • 5/s vs 200/s
Attack Queries

- Directed at L.root-servers.net
- IPv4 UDP
- Hit ham01.l.root-servers.org node
- Querying for: . /IN/ANY?
- Typical packet:

  192.0.2.1.54321 > 199.7.83.42.53: 123+ [1au] ANY? . (28)
Attack Query Distribution

- Baseline, ~150 qps
- 1 source, ~370 qps
- 3 sources, ~2900 qps
- 1 source, ~220 qps

Sunday, 12 May, 13
Responses with ~45k queries / 5 minutes

- Queries
- No RRL
- RRL (defaults)
- RRL (redbarn)

BIND9
BIND10
Knot DNS
NSD3
NSD4
RRL Drops with ~45k queries / 5 minutes

- BIND9
- BIND10
- Knot DNS
- NSD3
- NSD4

Queries: 0, 12500, 25000, 37500, 50000
RRL (defaults)
RRL (redbarn)
RRL Slips with ~45k queries / 5 minutes

- BIND9
- BIND10
- Knot DNS
- NSD3
- NSD4

Queries vs. RRL (defaults) vs. RRL (redbarn)
Responses with ~900k queries / 5 minutes

- BIND9
- BIND10
- Knot DNS
- NSD3
- NSD4

Queries: 0, 250000, 500000, 750000, 1000000

No RRL, RRL (defaults), RRL (redbarn)
RRL Drops with ~900k queries / 5 minutes

- Total Queries
- /IN/ANY?
- RRL (defaults)
- RRL (redbarn)

- BIND9
- BIND10
- Knot DNS
- NSD3
- NSD4

Monday, 12 May, 13
RRL Slips with ~900k queries / 5 minutes

- BIND9
- BIND10
- Knot DNS
- NSD3
- NSD4

Total Queries
./IN/ANY?
RRL (defaults)
RRL (reldbarn)
Conclusions?

• For this very small sample the different RRL implementations seem pretty similar
  • Redbarn RRL does more when the traffic level is low
  • Less difference as traffic ramps up
Further Work

• Repeat testing with more attack data
  • Got some you can push to OARC?
• Repeat testing with synthesized attack data
• Look at what impact running RRL has on other aspects of operation, RAM/CPU usage, etc
• Publish method and more results
Questions?
dave.knight@icann.org