Using dnstap with BIND

11 May 2016

ISC

Logistics

- Webinar is scheduled for 1 hour
- This session will be recorded and posted at http://www.isc.org/webinars
- Participants are muted to improve audio quality for everyone.
- We want questions! Please enter into the WebEx Q&A tab
  - The presenter may defer some questions until the end of the presentation

Presenters

Eddy Winstead

ISC

Senior Sales Engineer

Robert Edmonds

Fastly

Software Engineer

author of dnstap

Why dnstap?

Historically, our query logging options were:

- Query logging within the nameserver
- Packet capture external to the nameserver

Query logging

Good, but…

- We get query info but not response
- Significant performance impact

Query logging

parsing, formatting to text & shipping off to syslog or a file is quite pricey

External Packet Capture

Better performance, but…

- Additional infrastructure/processing
- Must deal with things the DNS server & networking stack already handle:
  - Discarding spoofed UDP packets
  - UDP fragments/TCP stream reassembly
  - UDP checksum verification

dnstap

dnstap

- Flexible, structured binary log format for DNS software
- Protocol Buffers logging for DNS software.

  https://developers.google.com/protocol-buffers/

"Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data."

dnstap

Add a lightweight **message duplication** facility directly into the DNS server

- Verbatim wire-format DNS messages with context.

ISC

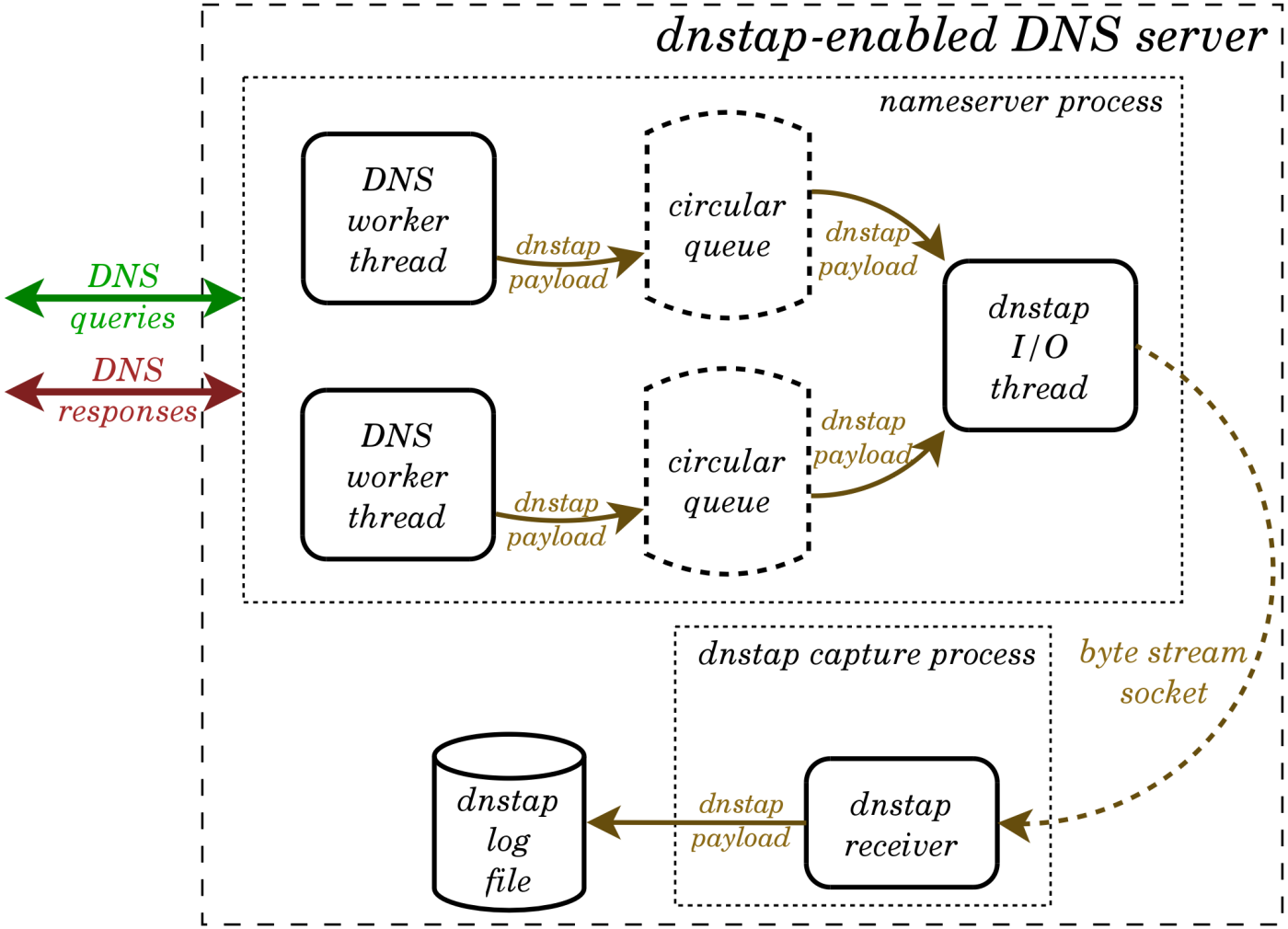
dnstap

Use fast logging implementation that doesn't degrade performance.

- Circular queues
- Asynchronous, buffered I/O
- Prefer to **drop** log payloads instead of **blocking** the server under load.

ISC

dnstap architecture



logging while diverse

dnstap implementations:

- Unbound
- Knot 2.x
- BIND
- NSD (planned)
- PowerDNS (planned)

Configuring BIND to use dnstap

- Install required library packages
- Build BIND with --enable-dnstap
- Add dnstap options to named.conf

Required library packages

All are available via github:

https://github.com/google/protobuf
https://github.com/protobuf-c/protobuf-c
https://github.com/farsightsec/fstrm

ISC

BIND versions with dnstap

- BIND with dnstap is available via the public source tree and in the stable preview version, 9.9.8-S4
- Included in our downloadable tarballs with the release of 9.11.0 (alpha available now, stable/final available in August)

Building BIND

- ./configure --enable-dnstap
- make
- make install

dnstap options for named.conf

- dnstap: bracketed list of message types to be logged
- dnstap-output: where to send
- dnstap-identity: optional id string to send
- dnstap-version: optional version string to send

dnstap option (from the ARM)

- The dnstap option is a bracketed list of message types to be logged. These may be set differently for each view. Supported types are client, auth, resolver, and forwarder. Specifying type all will cause all dnstap messages to be logged, regardless of type.
- Each type may take an additional argument to indicate whether to log query messages or response messages; if not specified, both queries and responses are logged.

named.conf examples

```
dnstap {auth; resolver query;} ;
dnstap-output unix "/var/run/bind/dnstap.sock”;
```

Or, to output to a file:

```
dnstap { all; } ;
dnstap-output file "/var/tmp/example.dnstap”;
```

logging to unix socket…

- Start fstrm_capture prior to starting BIND

```
# fstrm_capture –t protobuf:dnstap.Dnstap -u /var/
run/bind/dnstap.sock -w /var/tmp/example.dnstap
```

dnstap-read utility

Yay!, we have data!... how to see this with human eyes?

```
# /usr/local/sbin/dnstap-read example.dnstap

11-May-2016 05:34:21.900 CQ 172.20.6.224 UDP 29b www.isc.org/IN/A
11-May-2016 05:34:21.901 RQ 192.203.230.10 UDP 28b ./IN/NS
11-May-2016 05:34:21.903 RR 192.203.230.10 UDP 913b ./IN/NS
11-May-2016 05:34:21.904 RQ 198.41.0.4 UDP 47b ord.sns-pb.isc.org/IN/A
11-May-2016 05:34:21.901 RQ 192.203.230.10 UDP 40b www.isc.org/IN/A
11-May-2016 05:34:21.903 RR 192.203.230.10 UDP 569b www.isc.org/IN/A
11-May-2016 05:34:21.903 RQ 198.41.0.4 UDP 28b ./IN/NS
11-May-2016 05:34:21.904 RR 198.41.0.4 UDP 913b ./IN/NS
11-May-2016 05:34:21.904 RQ 198.41.0.4 UDP 48b sfba.sns-pb.isc.org/IN/A
```

dnstap-read utility

```
# /usr/local/sbin/dnstap-read -p example.dnstap

11-May-2016 05:34:21.900 CQ 172.20.6.224 UDP 29b www.isc.org/IN/A
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:   7592
;; flags: rd; QUESTION: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.isc.org.                    IN      A

11-May-2016 05:34:21.901 RQ 192.203.230.10 UDP 28b ./IN/NS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:  15909
;; flags:; QUESTION: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;.                               IN      NS

11-May-2016 05:34:21.903 RR 192.203.230.10 UDP 913b ./IN/NS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:  15909
```

dnstap-read utility

```
# /usr/local/sbin/dnstap-read -y example.dnstap

type: MESSAGE
identity: test-box
version: BIND 9.9.9-S1b2
message:
  type: RESOLVER_QUERY
  query_time: !!timestamp 2016-05-11T12:03:50Z
  socket_family: INET
  socket_protocol: UDP
  response_address: 192.203.230.10
  response_port: 53
  query_message: |
    ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:  25610
    ;; flags:    ; QUESTION: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
    ;; OPT PSEUDOSECTION:
    ; EDNS: version: 0, flags: do; udp: 4096
    ;; QUESTION SECTION:
```

Known Concern: Rotation!

- Ability to rotate dnstap files automatically as with current BIND logging: Coming in the next releases (SP + 9.11.0)

- fstrm modifications in progress

plans for dnstap

References

- http://dnstap.info
- http://lists.redbarn.org/mailman/listinfo/dnstap
- https://kb.isc.org/article/AA-01342/0/Using-DNSTAP-with-BIND-9.11.html

Thank You!

www.isc.org

info@isc.org