KNOT RESOLVER

Why is it different?

Petr Špaček • petr.spacek@nic.cz • 2017-10-25

CZ DOMAIN REGISTRY

What is Knot Resolver?

- Open-source DNS Resolver (GPLv3+)
- Follows DNS and DNSSEC standards
  - EDNS(0)
  - Automated Trust Anchor Management
  - ECDSA support
  - Negative Trust Anchors
  - Query Name Minimization

- *… so does BIND, Unbound, and others.*

Why is Knot Resolver different?

- Shared-nothing architecture
- 21st century features on by default
- Platform for building recursive DNS service!

Shared-nothing architecture

- SO_REUSEPORT
- Persistent shared cache – LMDB (optional)
- Crash! (definitely optional)

```
$ top
PID %CPU %MEM COMMAND
479 16.6  0.1 kresd
483 15.7  0.1 kresd
484 15.6  0.1 kresd
485 15.7  0.1 kresd
```

Scaling up: SO_REUSEPORT

- Start more processes

```
$ kresd -f 4
```

- under Systemd

```
$ systemctl mask kresd*.socket
$ systemctl edit kresd.service
[Unit]
RefuseManualStart=false
[Service]
ExecStart=/usr/sbin/kresd -c
/etc/kresd/config -f 4
```

Scaling up: shared cache

- **Configuration**

```
cache.size = 100 * MB
cache.storage = 'lmdb:///var/run/kresd/'
```

- **Monitoring**

```
$ ls -lh
… 100M … data.mdb

$ du -h data.mdb
60K data.mdb
```

21st century features

- **Enabled by default**
  - Query Name Minimisation (RFC 7814)
  - 0x20 Randomisation (`eXamPLE.cOm.`)
  - Run-time reconfiguration
  - Aggressive use of DNSSEC-validated Cache (RFC 8198, v2.0+)

- Available
  - Automatic DNSSEC bootstrap, DNS64, DNS "firewall", REST API, web interface, ...

Run-time reconfiguration

```
$ socat - UNIX-CONNECT:/run/kresd/control

> stats.list()
[answer.nxdomain] => 93148
…

> modules.load('policy')
> policy.add(
    policy.suffix(
      policy.FORWARD(
        {'2001:DB8::1', '192.0.2.1'}),
      {todname('example.')}))
```

Platform for recursive DNS service

- State machine
- Wide open!
- Lua, C, Go modules
- (ad 21st century features on by default …)
  - Config snippet

    ```
    modules = { 'workarounds < iterate' }
    ```
  - Shame list in `modules/workarounds/workarounds.lua`

Workarounds module: selectors

- modules/workarounds/workarounds.lua
- ```
  -- unfinished PTR query?
  if qry.stype ~= kres.type.PTR
     or bit.band(state,
       bit.bor(kres.FAIL, kres.DONE)) ~= 0
     then return state end
  ```
- ```
  -- zone cut?
  if qry.flags.AWAIT_CUT
     or qry.ns.name == nil
          then return state end
  ```

Workarounds module: hacks

- `modules/workarounds/workarounds.lua`
- ```
  -- fixes for problematic servers:
  -- (1) rdnsN.turktelekom.com.tr.
  if string.sub(name, 6) ==
                  '.turktelekom.com.tr.' then
     qry.flags.NO_0X20 = true
     qry.flags.NO_MINIMIZE = true
  end
  ```
  ```

DNS64 module: configuration

- `modules/dns64/dns64.lua`
- ```
  function mod.config (confstr)
      if confstr == nil then return end
  -- mod.proxy is module's variable!
      mod.proxy = kres.str2ip(confstr)
      if mod.proxy == nil then
          error('not a valid address')
      end
  end
  ```
  ```

DNS64 module: selectors

- `modules/dns64/dns64.lua`
-
```
is_nodata = (pkt:rcode() == NOERROR)
            and (#answer == 0)
if pkt:qtype() == AAAA and is_nodata
   and pkt:qname() == qry:name() -- CNAME?
   and (qry.flags.RESOLVED
        and qry.parent == nil) -- internal?
then
   extraFlags.DNS64_MARK = true -- mark
   -- get me an A!
   req:push(pkt:qname(), kres.type.A,
            kres.class.IN, extraFlags, qry)
```

DNS64 module: data transformation

- `modules/dns64/dns64.lua`
- ```
  -- query marked by selector?
  if qry.flags.DNS64_MARK then
     section = ffi.C.knot_pkt_section(pkt,
                                      kres.section.ANSWER)
     for i = 1, section.count do
        -- bit-play with addresses here
        -- add AAAA RRset to this answer
     end
  ```
- (use Lua instead of C FFI)
  ```

HTTP/2 API module

- `modules/daf/daf.lua`
- ```
  function M.config(conf)
      http.endpoints['/daf'] =
          {'application/json', api, publish}

  function api(header, stream)
      m = header:get(':method')
      if m == 'GET' then
          path = h:get(':path')

  -- do whatever you want here
  ```
  ```

Why is Knot Resolver different?

- Shared-nothing architecture
- 21st century features on by default
- **Platform for building recursive DNS service!**

Thanks to Ondřej Surý!

Stay tuned for Knot news!

lead by
Daniel Salzman

lead by
Petr Špaček

Knot news for October 2017

lead by
Daniel Salzman

lead by
Petr Špaček

- **Knot DNS 2.6**
- Automatic DNSSEC algorithm rollover
- In-line signing on slave

- **Knot Resolver 2.0**
- RFC 8198 aka Aggressive Use of DNSSEC-Validated Cache