OMG! A DNS Firewall!

Powerful DNS Filtering in Knot Resolver

Ondřej Surý • ondrej.sury@nic.cz • 27. 10. 2016

CZ.NIC | SPRÁVCE DOMÉNY CZ

Contents

- Knot Resolver
- HTTP/2 Interface
- DNS Firewall
- Demo running (and restarts every 2 minutes):
  - DNS: dig @demo.knot.dns.rocks
  - Web Interface: https://demo.knot.dns.rocks:8053

Knot Resolver

- Platform for building recursive DNS service
- Open-source DNS Resolver (GPLv3+) with DNSSEC support
- Written in C and LuaJIT
- Small daemon with dynamic configuration in Lua
- Scriptable (Lua) and extensible (Lua, C modules)
- No internal threading, asynchronous IO, scales by self-replication

Knot Resolver History

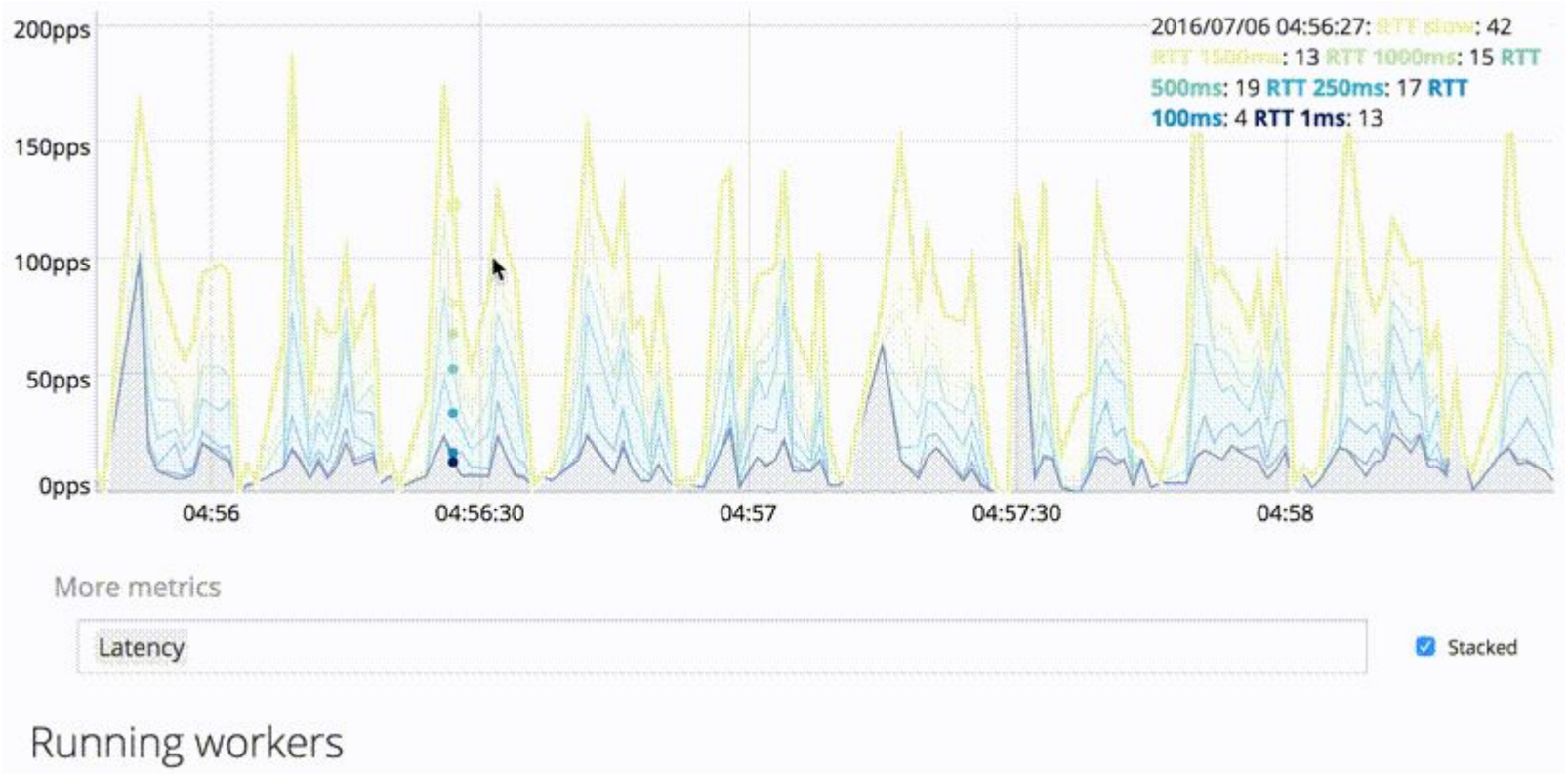
- Knot Resolver 1.0.0 – May 2016
  - https://indico.dns-oarc.net/event/22/session/3/contribution/1/material/slides/1.pdf
- Knot Resolver 1.1.0 – August 2016
  - DNS over TLS
  - Socket Activation
  - DNS Cookies
  - **HTTP/2 Interface**
  - **DNS Firewall**

HTTP/2 Interface

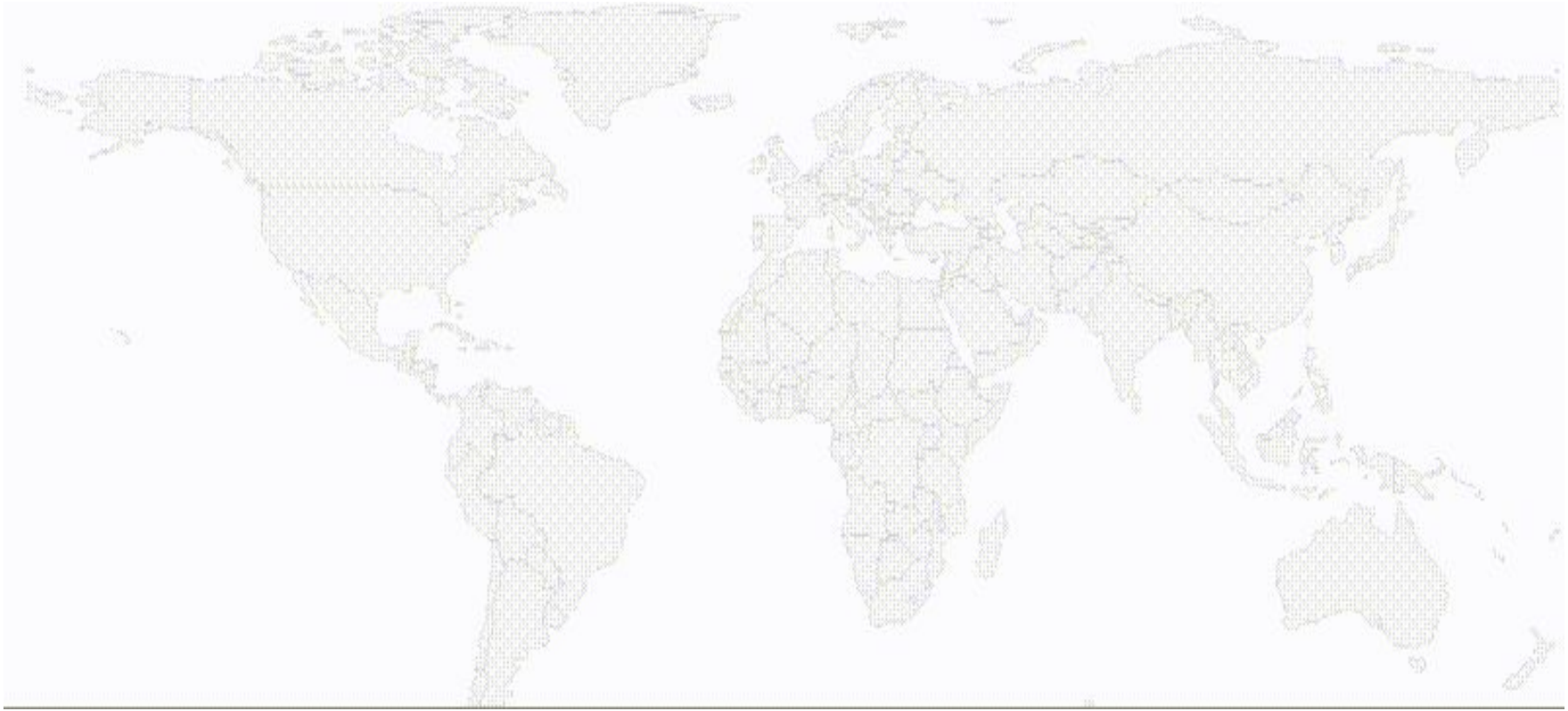
HTTP/2 Interface

- Web interface
- RESTful interface
  - Simple statistics
  - Prometheus pull-style metrics
- Built with lua-http and couple of JavaScript libraries

HTTP/2 Interface – Metrics



HTTP/2 Interface – GeoIP Information



DNS Firewall vs Access Lists

Access Lists

- Rules:
  - IP address based rules
- Actions:
  - Views
  - Zones (stubzones)
  - Forward
  - DENY, PASS

DNS Application Firewall

- Rules
  - IP address based rules
  - QNAME based rules
  - And/or logic
  - More rules easily implemented (in Lua)
- Actions (Policies)
  - PASS, DENY, DROP
  - TC
  - FORWARD
  - MIRROR
  - REROUTE/REWRITE

DAF Functions

- daf.rules
  - List the current rules table
- daf.add ‘<rule>’
- daf.get #no
- daf.delete #no
- daf.disable #no
- daf.enable #no

Rule: <selector> [and|or <selector>...] <action>

SELECTOR

- QNAME = <qname> #Exact match
- QNAME ~ <qname> #Lua pattern match
- SRC = <ipaddr> #Source address
- DST = <ipaddr> #Destination address

ACTION

- PASS
- DENY – return NXDOMAIN)
- DROP – (return SERVFAIL)
- REROUTE – (rewrite all IPs)
- REWRITE – (rewrite specific records)
- MIRROR – mirror the query
- FORWARD – forward query
- TRUNCATE – send TC bit

Rules examples

- Block all queries to ripe.net

daf.add('qname = ripe.net DENY')

- Drop all queries to <random>.knot-dns.cz

daf.add('qname ~ %w+.knot-dns.cz DROP')

Rewrite IPs (f.e. with known malware)

- Rewrite specific address to localhost

daf.add('src = 127.0.0.0/8 reroute 193.0.6.139-127.0.0.1')

- Rewrite subnet to different subnet

daf.add('src = 127.0.0.0/8 reroute 193.0.6.0/24-192.168.0.0')

- Rewrite specific name

daf.add('src = 127.0.0.0/8 rewrite ripe73.ripe.net A 127.0.0.1')

Query mirroring (for analysis, …)

daf.add('qname ~ %w+.example.com mirror 127.0.0.2')

daf.add('qname ~ example-%d.com mirror 127.0.0.3@5353')

Forward

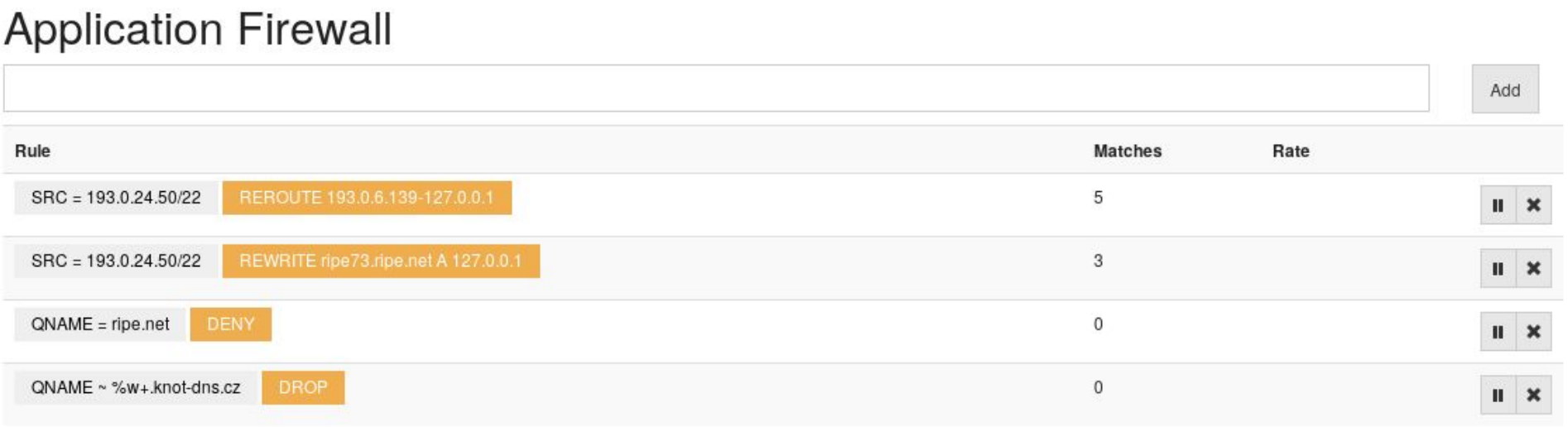
- Forward certain clients to different resolver

```
daf.add('src = 127.0.0.1/8 forward 127.0.0.1@5353')
```

- Forward subzone to different resolver

```
daf.add('qname ~ %w+.office.example.com forward 10.10.0.1')
```

Web Interface



RESTFul Interface

```
curl -s -X GET http://localhost:8053/daf | python -m json.tool
[
    {
    "active": true,
    "count": 0,
    "id": 1,
    "info": "qname = example.com deny"
    },
    {
    "active": true,
    "count": 0,
    "id": 2,
    "info": "qname ~ %w+.example.com AND src = 192.0.2.0/24 deny"
    }
]
```

RESTful Interface

- Add new rule
  curl -s -X POST -d "src = 127.0.0.1 pass" http://localhost:8053/daf
- Get rule <id>
  curl -s -X GET http://localhost:8053/daf/<id>
- Modify rule <id>
  curl -s -X PATCH http://localhost:8053/daf/1/active/false
- Delete rule <id>
  curl -s -X DELETE http://localhost:8053/daf/1

Questions?

Ondřej Surý • ondrej.sury@nic.cz • 27. 10. 2016