lieter_
PowerDNS

pieterlexis
PowerDNS

Dynamic answer generation with Lua

Pieter Lexis
Senior PowerDNS Engineer

February 2, 2019

Introduction

whoami

Pieter Lexis

- “Senior PowerDNS Engineer” at PowerDNS
- C++/Python/Go Developer
- System admin as well
- Packaging (RPM/DEB) wizard
- Build and test automation for the above

DNS, Lua and LUA-records

DNS is Mostly Static

- DNS round-robin $\neq$ real loadbalancing
- No failover for non-SRV services[1]
- No "specific answer" per requestor
- No real way to dynamically answer[2]

[1] like HTTP

[2] "Stupid DNS tricks"

Existing Solutions

- Many aaS vendors have proprietary solutions
  - Route 53 alias records
  - Cloudflare CNAME flattening
  - CNAME flattening at apex/ALIAS/ANAME
- PowerDNS has non-portable solutions
  - GeoIP backend
  - Remote backend
  - Pipe backend
  - Lua backend
- Bind has GeoIP features in 9.10 (view-like)

We'd like something that...

- Can generate answers dynamically
- Exist in the zone-file
- Can be AXFR'd between different implementations
- Requires no changes in recursors

Our solution

```
@ IN SOA ( ns.a.example. h.a.example. 2018020101
           10800 3600 604800 3600 )
@ IN LUA A ( "ifportup(443,                        "
             "  {'192.0.2.15', '198.51.100.20'})" )

@ IN LUA AAAA ( "ifportup(443,                     "
                "  {'2001:DB8:1::3A',              "
                "   '2001:DB8:5AC::4'})            " )
```

LUA-RECORDS...

- Are small Lua scripts
- Live in the zone
- Processing happens at runtime
- Helper functions forcertain types (A, AAAA, TXT, CNAME, LOC, PTR)

What is Lua?

Lua is

"a powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description."

Why Lua?

- Already embedded in Recursor and dnsdist
- Small, “no batteries included”
- Has many language bindings[3]
- LuaWrapper can wrap C++ functions to Lua functions

[3] Lua can be embedded in Go, Python and even LaTeX

LUA-RECORDS IN ACTION

Variables

- `who` – Client IP address
- `ecswho` – Client IP address from EDNS Client Subnet
- `bestwho` – `ecswho` when it exists, `who` otherwise

Functions – Address records

- `pickrandom` – Return an IP randomly from a list
- `pickwrandom` – Return a random IP address based on address weight
- `pickwhashed` – Return an IP address based on weight, but sticky
- `pickclosest` – Pick the addresses 'closest' to `bestwho`

Functions – Other

- `view` – Implement views based on source addresses
- `latlon` – Returns GeoIP latitude-longitude for `bestwho`

pdns.conf

```
# Bind addrs
local-address=127.0.0.1
local-ipv6=::1
local-port=5300

# Enable some features we need
edns-subnet-processing=yes
enable-lua-records=yes

# Serve zones from the BIND backend
launch=bind
bind-config=./named.conf
```

named.conf

```
zone "example.nl" in {
  type native;
  file "example.nl.zone";
};

zone "10.in-addr.arpa" in {
  type native;
  file "10.in-addr.arpa.zone";
};

zone "8.b.d.0.1.0.0.2.ip6.arpa" in {
  type native;
  file "8.b.d.0.1.0.0.2.ip6.arpa.zone";
};
```

example.nl.zone

```
$ORIGIN example.nl.
@ IN SOA ns1.example.nl. hostmaster.example.nl. 1 2 3 4 5

@ IN LUA A "pickrandom({'1.2.3.4', '2.4.5.6', '3.4.5.6'})"

service IN LUA A (";if (netmask({'10.0.0.0/8'})) then "
                  "  return '10.4.5.6'                "
                  "else                               "
                  "  return '192.168.2.15'            "
                  "end                                ")

service2 IN LUA CNAME ( "view({                                  "
                        "{ {'192.0.2.0/24'}, {'system1.example.nl'} }, "
                        "{ {'10.0.0.0/24'}, {'system2.example.nl'} },   "
                        "{ {'0.0.0.0/0'}, {'system3.example.nl'} }     "
                        "}) " )

system1 IN LUA A ( " ifportup(80, {'127.0.0.1', '192.168.0.5'}) ")

system2 IN LUA A ( " ifportup(80, {'10.0.0.2', '192.168.0.5'}, {selector='pickclosest',
  ↪    backupSelector='random'}) ")

system3 IN A 192.168.2.3

txt IN LUA TXT ( "'Your IP address is ' .. bestwho:toString()")
```

pickrandom

```
@ IN LUA A "pickrandom({'1.2.3.4', '2.4.5.6',
 ↪  '3.4.5.6'})"
```

```
$ dig @127.0.0.1 -p5300 +norec +short example.nl A
3.4.5.6
$ dig [...] example.nl A
2.4.5.6
$ dig [...] example.nl A
2.4.5.6
$ dig [...] example.nl A
3.4.5.6
$ dig [...] example.nl A
2.4.5.6
```

IF/THEN/ELSE

```
service IN LUA A (";if (netmask({'10.0.0.0/8'})) then "
                  "  return '10.4.5.6'                 "
                  "else                                "
                  "  return '192.168.2.15'             "
                  "end                                 ")
```

```
$ dig [...] service.example.nl A
192.168.2.15

$ dig [...] service.example.nl A +subnet=10.0.0.0/8
10.4.5.6
```

view

```
service2 IN LUA CNAME ( "view({                                  "
                        "{ {'192.0.2.0/24'}, {'system1.example.nl'} }, "
                        "{ {'10.0.0.0/24'}, {'system2.example.nl'} },   "
                        "{ {'0.0.0.0/0'}, {'system3.example.nl'} }      "
                        "}) " )
```

```
system3 IN A 192.168.2.3
```

```
$ dig [...] service2.example.nl A
system3.example.nl.
192.168.2.3
```

view

```
service2 IN LUA CNAME ( "view({                                    "
                        "{ {'192.0.2.0/24'}, {'system1.example.nl'} }, "
                        "{ {'10.0.0.0/24'}, {'system2.example.nl'} },   "
                        "{ {'0.0.0.0/0'}, {'system3.example.nl'} }     "
                        "}) " )
```

```
system2 IN LUA A ( " ifportup(80, {'10.0.0.2', '192.168.0.5'}, {selector='pickclosest', backupSelector=
```

```
$ dig [...] service2.example.nl A +subnet=10.0.0.0/8
system2.example.nl.
192.168.0.5

$ dig [...] service2.example.nl A +subnet=10.0.0.0/24
system2.example.nl.
192.168.0.5

$ dig [...] service2.example.nl A +subnet=11.0.0.0/24
system3.example.nl.
192.168.2.3
```

view

```
service2 IN LUA CNAME ( "view({                                    "
                        "{ {'192.0.2.0/24'}, {'system1.example.nl'} }, "
                        "{ {'10.0.0.0/24'}, {'system2.example.nl'} },   "
                        "{ {'0.0.0.0/0'}, {'system3.example.nl'} }     "
                        "}) " )
```

```
system1 IN LUA A ( " ifportup(80, {'127.0.0.1', '192.168.0.5'}) ")
```

```
$ dig [...] service2.example.nl A +subnet=192.0.2.0/24
system1.example.nl.
127.0.0.1

$ dig [...] service2.example.nl A +subnet=192.0.2.0/24
system1.example.nl.
192.168.0.5
```

Functions – PTR records

- `createReverse` – Generate default hostnames for in-addr.arpa addresses
- `createReverse6` – Generate default hostnames for ip6.arpa addresses
- `createForward` – Generate A record from a default hostname
- `createForward6` – Generate AAAA record from a default hostname

10.in-addr.arpa.zone

```
$ORIGIN 10.in-addr.arpa.
@ IN SOA ns1.example.nl. hostmaster.example.nl. 1 2 3 4
 ↪  5

* IN LUA PTR
 ↪  "createReverse('%1%.%2%.%3%.%4%.hosts.example.nl.')"
*.1 IN LUA PTR "createReverse('%5%.hosts.example.nl.')"
*.2 IN LUA PTR "createReverse('%6%.hosts.example.nl.')"
```

createReverse

```
* IN LUA PTR "createReverse('%1%.%2%.%3%.%4%.hosts.example.nl.')"
*.1 IN LUA PTR "createReverse('%5%.hosts.example.nl.')"
*.2 IN LUA PTR "createReverse('%6%.hosts.example.nl.')"
```

```
$ dig [...] 12.4.5.10.in-addr.arpa PTR
10.5.4.12.hosts.example.nl.

$ dig [...] 2.0.1.10.in-addr.arpa PTR
10-1-0-2.hosts.example.nl.

$ dig [...] 2.0.2.10.in-addr.arpa PTR
0a020002.hosts.example.nl.
```

8.b.d.0.1.0.0.2.ip6.arpa.zone

```
$ORIGIN 8.b.d.0.1.0.0.2.ip6.arpa.
@ IN SOA ns1.example.nl. hostmaster.example.nl. 1 2 3 4
 ↪  5

* IN LUA PTR "createReverse6('%33%.hosts.example.nl.')"
```

createReverse

```
* IN LUA PTR "createReverse6('%33%.hosts.example.nl.')"
```

```
$ dig [...] -x 2001:db8:ba:34::2
2001-db8-ba-34--2.hosts.example.nl.
```

More Information

Security of LUA-records

- No sandboxing at the moment
- LUA records can be enabled globally or per-domain[4]
- Use more CPU cycles than regular records
- Limited to 1000 instructions by default
  `lua-records-exec-limit`

[4] ENABLE-LUA-RECORDS domain metadata

Current State

- Usage is still a bit rough
- Needs the GeoIP backend loaded for Geo-magic
- No pre-flight checks
- It works!

What's next?

- Release Authoritative Server 4.2.0
- Get experience with LUA records
- Polish the implementation
- Create a minimal set of useful functions
- Come up with a proper version 1 specification
- Get that version 1 specification in more implementations