Deckard

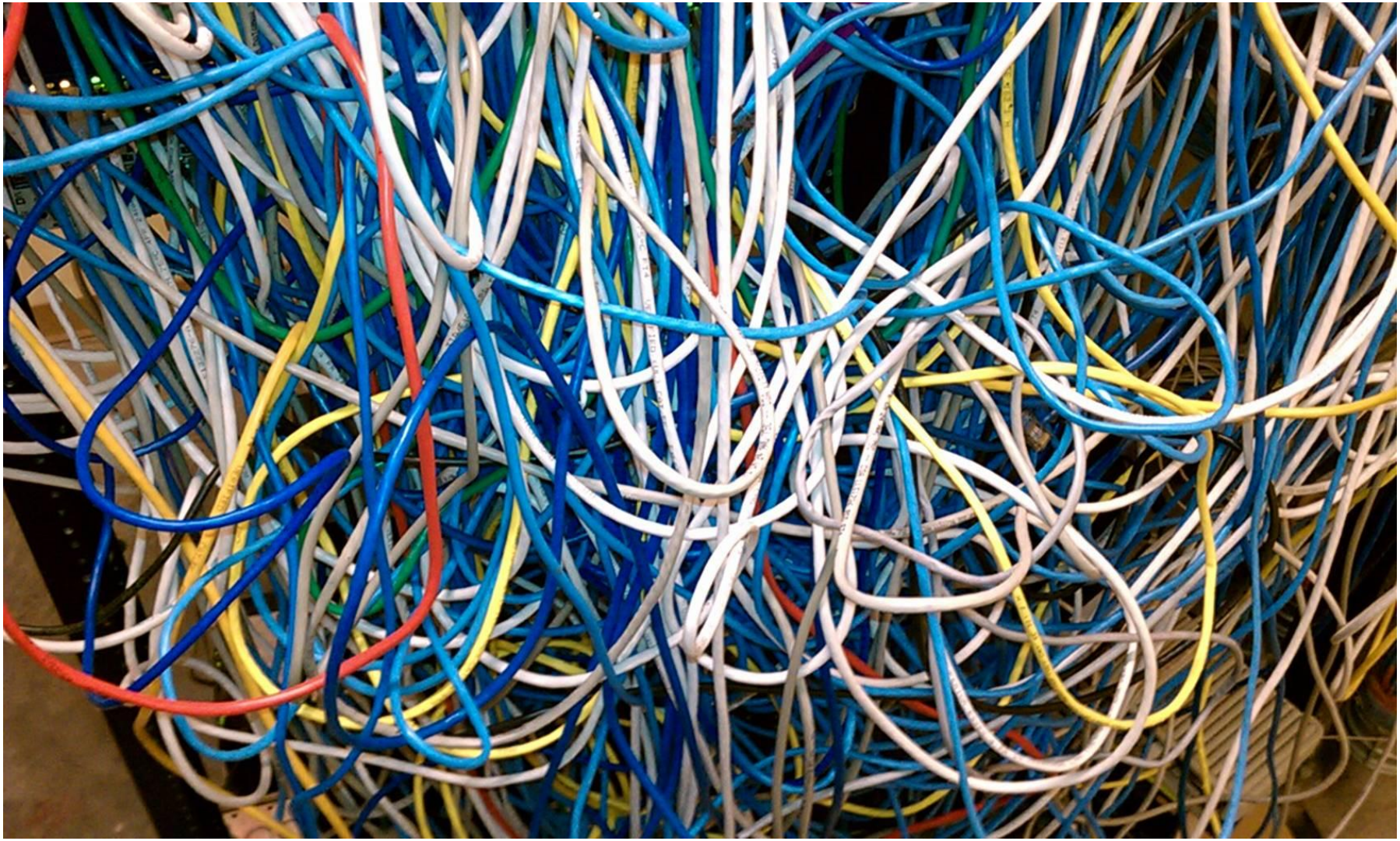
I've seen things you DNS people wouldn't believe.

Ondřej Surý • ondrej.sury@nic.cz • 19. November 2015

Testing DNS software

- Standards compliance
- “Real-world” compliance
- Repeatable
  - Test on “live” Internet?
  - Setup a complicated test lab?
  - Hooks in the code?
  - …

DNS software “test lab”



Deckard – a software “test lab”

- Developed during Knot DNS Resolver development
- Simulate everything on runtime
  - Creates a controlled environment
  - Wraps around “syscalls” and standard library functions
- The tests are fast
  - Testing is done in seconds
- Thus could be integrated into development cycle
  - Continuous integration

Deckard – a software “test lab”

- socket_wrapper
  - Creates fake network environment
  - Application can bind to privileged ports
  - Communicate with mocked servers
- libfaketime
  - Change the flow of the time
- Jinja2
  - For creating configuration from templates

Test scenarios

- Inspired by Unbound replay.h test cases
- Runs the **production** binary as a subprocess
- Redirects all network communication to UNIX sockets
- Declarative description of the environment
  - DNS server configuration
  - Network configuration
- Sequence of DNS messages
  - Queries to be asked
  - Answers to be given
  - Expectations about answers

Scenario example – Jinja2 config template

- {{SELF_ADDR}}
  - Address to bind to
- {{ROOT_ADDR}}
  - Fake root hints
- {{NO_MINIMIZE}}
  - QNAME Minimization
- {{TRUST_ANCHOR}}
  - DNSSEC Root Key

```
net = { '{{SELF_ADDR}}' }
modules = {'stats', 'policy', 'hints'}
hints.root({['k.root-servers.net'] = \
  '{{ROOT_ADDR}}'})
option('NO_MINIMIZE', {{NO_MINIMIZE}})
option('ALLOW_LOCAL', false)
trust_anchors.add('{{TRUST_ANCHOR}}')
verbose(true)
[...]
```

Scenario example – lame root

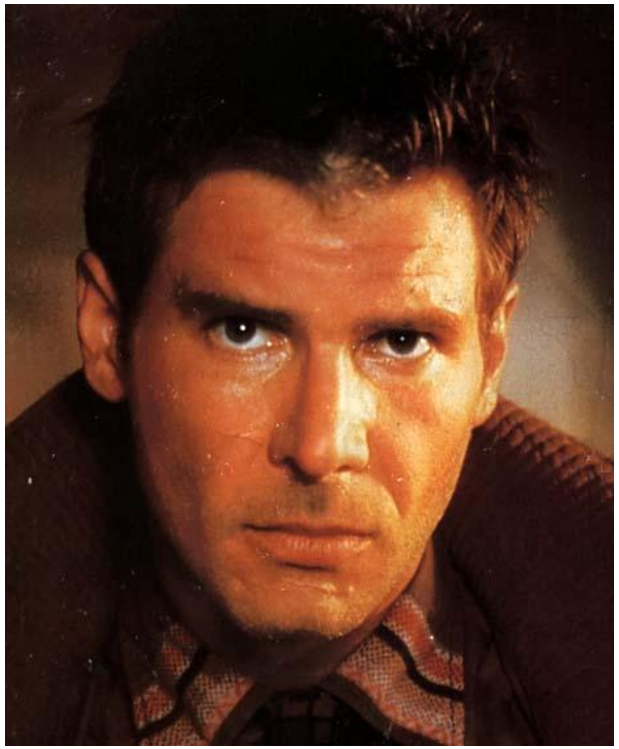
- RANGE m n – RANGE_END
  - Define a set of queries/answers used in STEPS m-n
- ENTRY_BEGIN – ENTRY_END
  - A description of DNS message
- MATCH
  - Which queries does trigger the answer
  - all, opcode, qtype, qname, …
- ADJUST
  - copy_id || copy_query
- REPLY <flag>|<rcode>
  - Set flags
  - Set RCODE
- SECTION <type> – END_SECTION
  - RR data to return in the DNS message
- STEP <n> <type>
  - QUERY | REPLY | CHECK_ANSWER | …
- RAW
  - Encoded binary content

```
CONFIG_END
SCENARIO_BEGIN Test iterative resolve with lame root.

; K.ROOT-SERVERS.NET.
RANGE_BEGIN 0 100
        ADDRESS 193.0.14.129
ENTRY_BEGIN
MATCH opcode qtype qname
ADJUST copy_id
REPLY QR RA SERVFAIL
SECTION QUESTION
. IN NS
ENTRY_END
RANGE_END

STEP 1 QUERY
ENTRY_BEGIN
REPLY RD
SECTION QUESTION
www.example.com. IN A
ENTRY_END

; recursion happens here.
STEP 10 CHECK_ANSWER
ENTRY_BEGIN
MATCH all
REPLY QR RD RA SERVFAIL
SECTION QUESTION
www.example.com. IN A
ENTRY_END

SCENARIO_END
```

Deckard – further development

- Free software, open-source
  https://gitlab.labs.nic.cz/knot/deckard
- Scenario guide and more scenarios:
  https://gitlab.labs.nic.cz/knot/deckard/blob/master/SCENARIO_GUIDE.rst
  https://gitlab.labs.nic.cz/knot/deckard/tree/master/sets
- More complicated scenario example:
  https://gitlab.labs.nic.cz/knot/deckard/blob/master/SCENARIO_EXAMPLE.rst
- You are welcome to participate
  - More test cases – for resolvers, authoritative DNS, and even DNS tools
  - More servers to test – Jinja2 templates for the configuration
  - More (and regular) testing

Questions?



Ondřej Surý • ondrej.sury@nic.cz • 19. November 2015