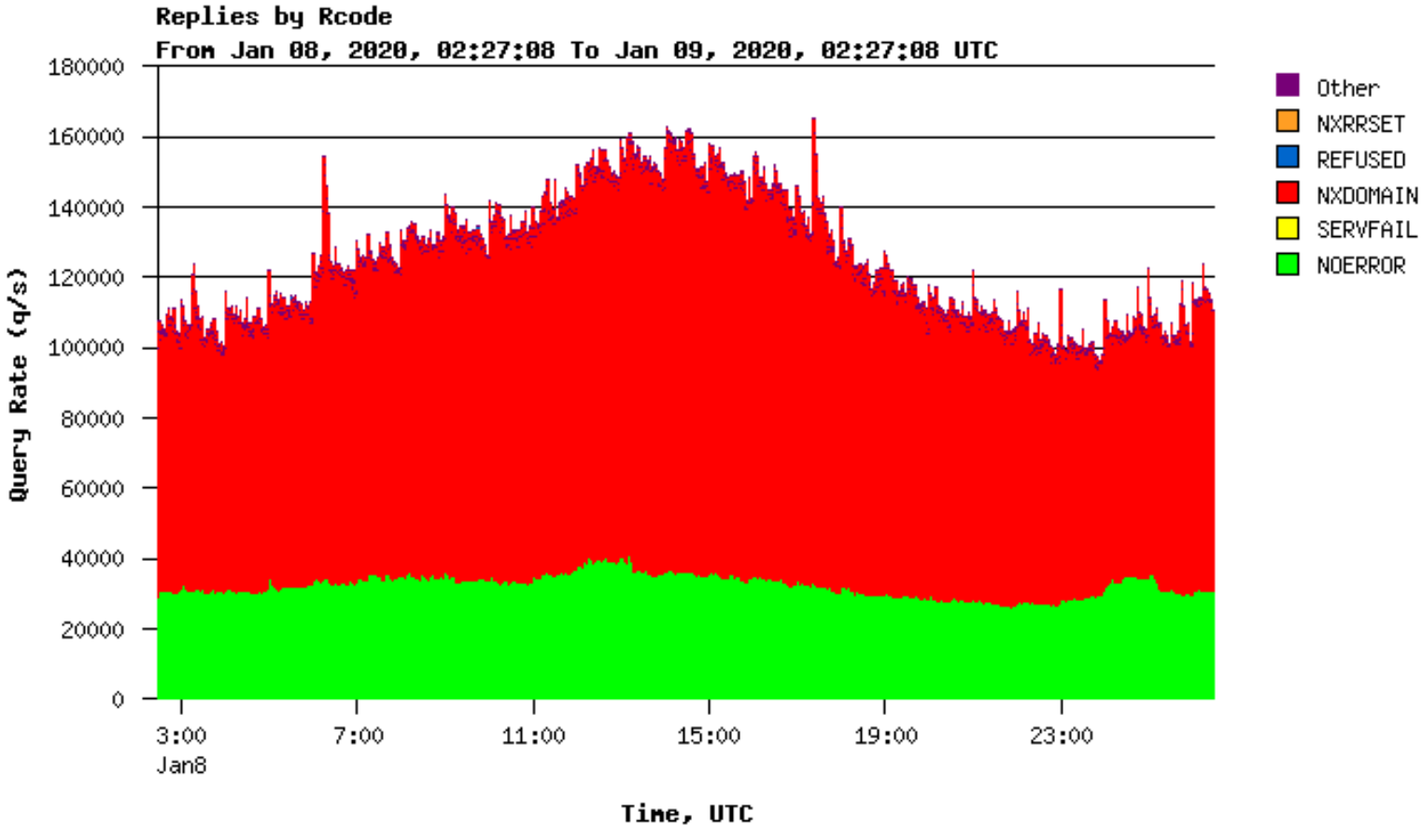
unwind(8) - florian@OpenBSD.org

- OpenBSD developer since 2012
  - author of slowcgi(8), slaacd(8) (cf. BSDCan 2018), rad(8), unwind(8), sysupgrade(8), ...
  - poked at things in the network stack
  - `1744 files changed, 229095 insertions(+), 337080 deletions(-)`
- Senior Systems Engineer @ RIPE NCC
  - BGP, DNS, ...
  - k.root-servers.net, ...



A Day in the Life of a Root Name Server

- The Problem and A Solution

- A recursive name server for every laptop

- Opportunistic DNSSEC validation

- Captive-portal handling

- Adapt to local conditions...

- … no matter how harsh.

Previous approaches

- dhclient(8)
  - Just owns /etc/resolv.conf
  - Will get you past captive-portals
  - At the mercy of recursive name server operator
  - No DNSSEC
  - No privacy
  - Same address space

- static configuration
  - Tell dhclient to leave /etc/resolv.conf alone
  - Will likely not get you past captive-portals
  - Will not work in places where DNS is filtered
  - No DNSSEC
  - No privacy
  - Same address space

- run unbound(8) on localhost
  - Tell dhclient to leave /etc/resolv.conf alone
  - DNSSEC validation
  - Privacy: (DoT) if configured
  - Will likely not get you past captive-portals
  - Will not work in places where DNS is filtered
  - Different address space

- Others
  - systemd-resolved
  - openresolv / resolvconf
  - stubby
  - dnsmasq
  - dnscrypt-proxy
  - ...

Welcome unwind(8).

- Run unwind(8) on localhost
  - Tell dhclient to leave /etc/resolv.conf alone
  - DNSSEC validation
  - Privacy: (DoT) if configured
  - Will get you past captive-portals
  - Will work in places where DNS is filtered
  - Different address space

- Architecture
  - Three processes privileg seperated daemon
  - Design copied from bgpd, ospfd, rad, slaacd...
    - IPC over pipes
    - Config file & reload
    - Logging to syslog & stderr
    - Control tool
  - pledge(2) for restricted service operating mode
    - see www.openbsd.org/events.html

- Parent process
  - Starts everything up
  - Opens localhost 53/udp & passes to frontend
  - Asks dhclient & slaacd to send forwarders
  - `pledge("stdio rpath sendfd");`
  - Config parsing & reload

- Frontend process
  - `pledge("stdio unix recvfd");`
  - Receives queries on 53/udp
    - Parses query
    - Passes query to resolver
    - Sends back response
  - Handles control socket
  - Listens for interfaces going up / down
  - Listens for learned forwarders

- Resolver process
  - All the heavy DNS lifting
    - Keeps track of resolving strategies
    - Receives query from frontend
    - Hands query to "best" strategy
    - Sends answer back to frontend
  - `pledge("stdio inet rpath");`
  - `unveil("/etc/ssl/cert.pem", "r");`

Relax...

- Resolver: libunbound
  - Does most of the resolving for us
  - Local copy
    - Not a fork; kept in sync with upstream
    - We need to poke at some internals

- Resolver: Strategies (1/2)
  - Without config file:
    - recursor
    - dhcp / slaac forwarder
    - dhcp / slaac opportunistic DoT forwarder
    - dhcp / slaac stub forwarder
  - With config file:
    - DoT forwarders with authentication name
    - static forwarders
  - Default preference: DoT forwarder recursor dhcp stub

- Resolver: Strategies (2/2)
  - individual libunbound contexts
  - (except stub: asr(3) - libc asynchronous resolver)
  - shared cache
  - single threaded, plugged together with libevent

- Resolver: Health & quality checks (1/2)
  - `. IN SOA`
    - Known to exist
    - DNSSEC signed
  - Strategy state
    - `VALIDATING`
    - `RESOLVING`
    - `UNKNOWN`
    - `DEAD`

- Resolver: Health & quality checks (2/2)
  - Track query round trip time
  - Keep a (decaying) histogram per strategy
  - Calculate median RTT

```
1. recursor         validating,  50ms   4. DoT              validating, 150ms
2. forwarder        validating,  30ms   5. stub              resolving,   N/A
3. dhcp             validating,  70ms
```

histograms: lifetime[ms], decaying[ms]

	<10	<20	<40	<60	<80	<100	<200	<400	<600	<800	<1000	>
rec	15664	413	1784	1363	809	552	987	601	206	55	44	425
	11	0	2	2	0	0	1	0	0	0	0	0
forw	312	63	68	36	37	48	138	124	88	24	12	446
	0	0	0	0	0	0	0	0	0	0	0	0
dhcp	33	7	7	1	3	0	2	4	1	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
DoT	33	0	0	4	26	25	82	43	28	25	11	696
	0	0	0	0	0	0	0	0	0	0	0	0
stub	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0

- Resolving
  - Sort strategies
    - `VALIDATING` > `RESOLVING`
    - Median RTT
    - Prefered strategy skewed by 200ms
  - Pick best; call `ub_resolve_event()`
  - Wait median RTT ms; no answer:
    - Try next best strategy
    - Do **not** cancel running queries

Captive Portals & DNS breakage

- Captive Portals (1/5)
  - Unwind used to have an http checker...
  - ... but we couldn't agree on a built-in URL

- Captive Portals (2/5)
  - Everything blocked except dhcp forwarders
    - → use those

- Captive Portals (3/5)
  - Everything blocked except dhcp forwarders or DNS gets intercepted
  - `SERVFAIL` or `NXDOMAIN` with edns0
  - → use stub

- Captive Portals (4/5)
  - DNS open
  - 1st (or 2nd!) http redirect results in `NXDOMAIN`
  - Heuristic: Do not trust `NXDOMAIN`

    → use stub

- Captive Portals (5/5)
  - DNS open
  - Portal domain DNSSEC signed
  - No signature for portal host
  - Heuristic: Do not trust validation errrors

    → use stub

Future work

- Future work
  - Wifi vs 4g
  - DNSSEC validation too opportunistic
  - Dedicated TCP strategy
  - Track captive portal URL domains
  - don't use shared cache behind captive portals
  - DNS64

Questions?

Come on! Don't be shy!