The Last Millimetre

Remind me again why we are doing this DNSSEC stuff

Ron Aitchison

DNSSEC's Purpose

- Classic RFC Stuff
  - Authenticated - Authoritative source
  - Integrity - Data is unmodified
  - PNE - Negative responses correct
- So what....

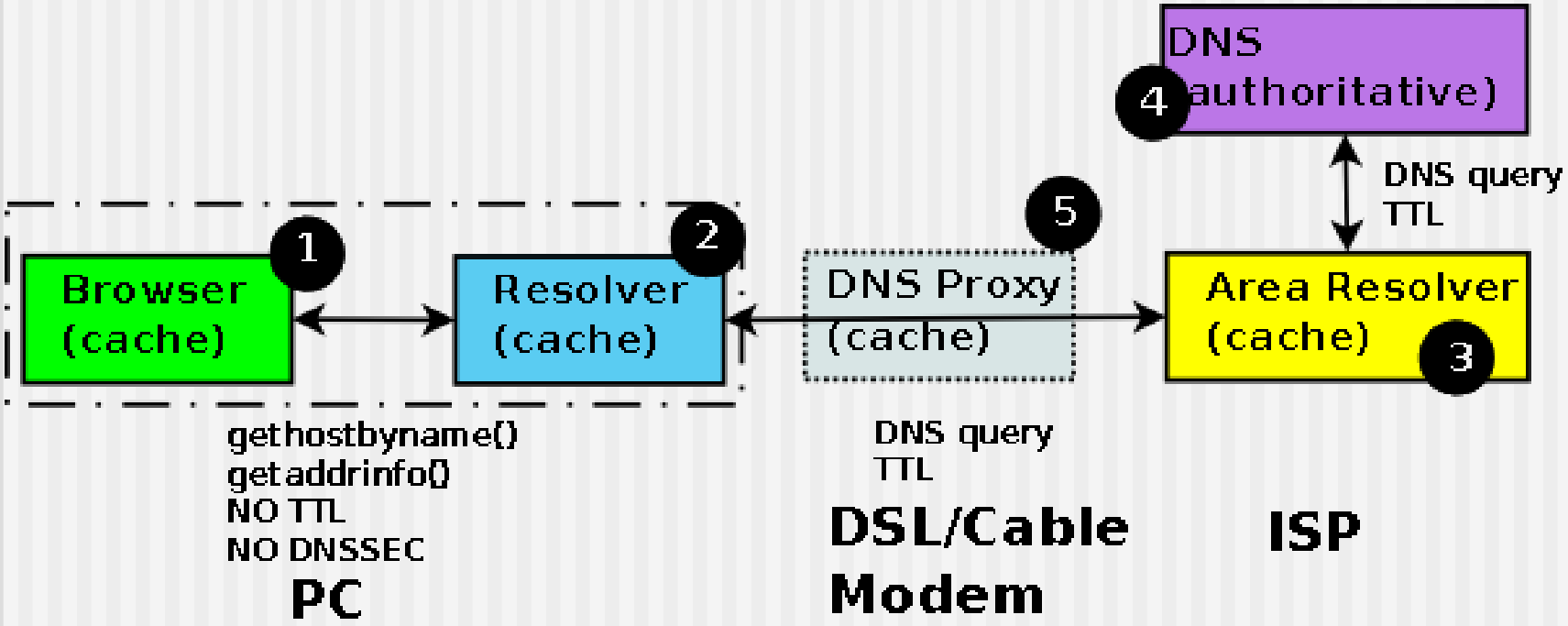
DNSSEC's Purpose

- Classic RFC Stuff Response
  - Authenticated - Data from authoritative source
  - Integrity - Data is unmodified
  - PNE - Negative responses correct
- So what....
  - 'cos applications can use the results
    - Browsers, Mail, LDAP clients etc.
- Which means...

Desktop DNSSEC Requirements

- DNSSEC-aware API
  - POSIX, MS
- DNSSEC-aware/validating stub (caching) resolver
- Root-distribution and rollover

DNSSEC API


DNS
authoritative)
4
DNS query
TTL
5
1
2
Browser
(cache)
Resolver
(cache)
DNS Proxy
(cache)
Area Resolver
(cache)
3
gethostbyname()
getaddrinfo()
NO TTL
NO DNSSEC
PC
DNS query
TTL
DSL/Cable
Modem
ISP

Desktop DNSSEC Requirements

- DNSSEC-aware API
  - POSIX (getaddrinfo), MS (getaddrinfo, DnsQuery).
- w/out DNSSEC-aware/validating stub resolver
- Multiple approaches so far:
  - draft-hayatnagarkar-dnsext-validator-api-09/libval (dnssec-tools)
    - val_getaddrinfo, no BOGUS, no TTL, configurable, manual trust anchor, sync/async
  - libunbound
    - ub_resolve, big answer (conflates insecure and indeterminate), sync/aync, no CD mode? Allows for bridge/wrapper function
  - libdns (isc)
    - DNSSEC aware getaddrinfo, only BOGUS (via gai_strerror), no TTL, manual trust anchor, sync

DNSSEC API - Outstanding

- Need a DNNSEC-aware API:
  - Standardized (IETF/POSIX-IEEE)
  - Works with and without DNSSEC
  - Primary status - OK/Fail (as today)
  - Auxiliary status: None, Secure (1 bit?)
  - But would be useful to solve the blind cache problem (TTL)

Desktop DNSSEC Requirements

- DNSSEC-aware API
  - POSIX, MS
- DNSSEC-aware/Validating Stub (caching) Resolver

Stub Resolver

- DNSSEC-aware:
  - Use AD, interprets results
  - Pretty simple - needs an API
  - No trust anchor
  - Area Resolver (DDoS)
- DNSSEC Validation:
  - Pretty complex - needs an API
  - Full resolver - (unbound/Bind)
  - use CD
  - Needs trust anchor
  - Distributed validation load

DNSSEC Validation Stub

- Full Resolver
  - Code Base exists (BIND/Unbound/others)
  - No Area Resolver (caching) usage
  - Mobile data volume
  - Every device is exposed
- Use CD
  - Area Resolver (caching) use
  - Mobile data volume lower
  - Code base changes (?)

DNSSEC-Stub - Outstanding

- DNSSEC-API
- All methods possible/exist
- Code changes/implementation
- May need to solve mass root-key problem:
  - Local validating stubs are orders of magnitude bigger (maybe)

Desktop DNSSEC Requirements

- DNSSEC-aware API
  - POSIX, MS
- DNSSEC-aware/Validating Stub (caching) Resolver
- root-key handling

root-key Handling

- Distribute with everything (OS, BIND, Unbound, etc.)
  - just like root ca certs
  - getting it via HTTPS is a joke
- Key-rollover
  - We need a constant test bed (DLV like approach?)
  - Stick 'em under root.arpa (test/emergency/next/backup, etc.) as DNSKEY RR or new RR type
  - rollover - fail to validate use the emergency key (5011'ish process)
  - Biggest problem is not key but algorithm rollover
  - Depending on approach area resolvers have cache
  - What's the attack vector for a compromised root-key
  - It's either too late or we have until 2027 (ish)
  - ICANN security/process vs X.509 CAs

root-key - Outstanding

- We need an in-band RFC process for root/algorithm change
  - RFC 5011 made a huge leap forward - we need to build on it

Desktop DNSSEC Requirements

- DNSSEC-aware API
  - Single API Standardization
- DNSSEC-aware Stub (caching) Resolver
  - Multiple approaches
  - Just code (!)
  - Mobile?
- Root-key distribution
  - initial with distribution (stub or OS)
  - use DNS tree to provide changes
  - algorithms changes are tough
  - X.509 root certs sets the bar

Thanks for your patience