IPv6 Whitelist Operations

Erik Kline, Google
The makings of a problem

- DNS resolution of AAAAs is the effectively the one and only control knob for enabling/disabling IPv6 traffic to a website.
- RFC 3596: "The IP protocol version used for querying resource records is independent of the protocol version of the resource records; e.g., IPv4 transport can be used to query IPv6 records and vice versa."
  - basically required...but it does break fate-sharing
- How to restore some semblance of fate-sharing?
  - BIND's disable-aaaaa-on-v4-transport
  - draft-vandergaast-edns-client-ip
  - temporary use of "whitelisting"
Why whitelist?

To express the quality of working IPv6

- Fate-sharing for DNS only indicates that a ~512 byte packet wasn't dropped
- Want users to have the best possible experience
  - what is the impact of 0.05+% of users experiencing high latency or even not reaching the site at all?
- Not all IPv6 connectivity is equal
  - an AS may have worse IPv6 redundancy than IPv4
- Not all IPv6 networks are equally well supported
  - some operators may not want the IPv6 traffic (yet)
Exempli gratia

Normally, if a DNS resolver requests an IPv6 address for a Google web site, it will not receive one…

…but a DNS resolver in the Google over IPv6 "whitelist" will receive an IPv6 address, and its users will be able to connect to Google web sites using IPv6.

http://google.com/ipv6/
For each request:

1. Receive a list of resolvers and/or prefixes
2. Attempt to verify the requester owns/operates said prefixes
3. Convert to ASN(s), complete list of IPv4 and IPv6 prefixes
4. Verify mutual IPv6 connectivity is not worse than IPv4:
   - routing table comparison
   - test pMTUd
   - <insert other tests du jour>
5. Record commitment to production-quality operations
6. Possibly coordinate go-live time:
   - try to find a light traffic time
   - deal with timezone issues
   - coordinate handling of brokenness reports with NOCs
7. Possibly deal with emergency revert requests
8. ...iterate...
A different approach

- For each resolver: **signal readiness** to receive AAAAs
  - `_aaaa.1.2.0.192.in-addr.arpa. 1W IN TXT "ok"

- Actively **monitor** IPv6 traffic, trouble reports, and brokenness metrics
- Debug and **iterate**
What it is

- A method to explicitly signal readiness (or lack thereof) to receive AAAAs
- Uses "reverse DNS" delegations for loose verification of operational ownership
- Optionally uses TTLs to express desired lifetimes
  - ...but operational reality may trump this
- Pretty simple, in the common case, for operators
What it is not

- A membership-restricted club
- 100% automated and maintenance-free
- Equally handled by all providers
- Perfect
- A long-term solution
Syntax

Fairly straightforward, vis.:

_aaaa.1.2.0.192.in-addr.arpa. 1W IN TXT "ok"

;_aaaa.*.2.0.192.in-addr.arpa. 1W IN TXT "!ok"

_aaaa.1.[...].8.b.d.0.1.0.0.2.ip6.arpa.
    5D IN TXT "!ok"
Content provider-side processes

1. Log resolver IP addresses
2. Background lookups of "reverse DNS" names for TXT records with a specified format
3. Merge results into white- & blacklists, optionally with TTLs
   - remove (or blacklist) formerly whitelisted resolvers now opting out or no longer listing TXT records (expired)
   - impact analysis of proposed new whitelist entries
   - add or discard as determined by analysis
   - update running nameservers with new config
4. GOTO 1
Limitations

- Implementation (software and processes) may be a non-trivial effort
- Update timeliness not guaranteed
- Results of impact analysis opaque to requester
  - ...and privacy requirements hamper cooperation
- Does not necessarily allow for pair-wise opt-out or opt-in (i.e. it's all participating providers serve AAAAs or none do)
  - extended syntax makes this possible
  - ...but operational reality may trump this
Questions?

ipv6whitelist.org