DNS Evolution

Geoff Huston AM
APNIC Labs

https://xkcd.com/1361/

Why pick on the DNS?

The DNS is **used by everyone and everything**

- Because pretty much everything you do on the net starts with a call to the DNS
- If we could see your stream of DNS queries in real time we could easily assemble a detailed profile of you and your interests and activities - as it happens!

Why pick on the DNS?

The DNS is very **easy** to **tap** and **tamper**

- DNS queries are open and unencrypted
- DNS payloads are not secured and tampering cannot be readily detected
- DNS responses are predictable and false answers can be injected

Why pick on the DNS?

The DNS is **hard for users to trace**

- Noone knows exactly where their queries go
- Noone can know precisely where their answers come from

What are we doing about this?

I'd like to look at this question by grouping our responses into three areas of activity:

1. Adding authenticity to the DNS
2. Increasing the reliance on the DNS for application level rendezvous functions
3. Plugging DNS information leaks

1. **Adding authenticity to the DNS**
2. Increasing the reliance on the DNS for application level rendezvous functions
3. Plugging DNS information leaks

How can you trust the DNS answer?

- Send your query to the “right” IP address and you will get the “right answer!

Or

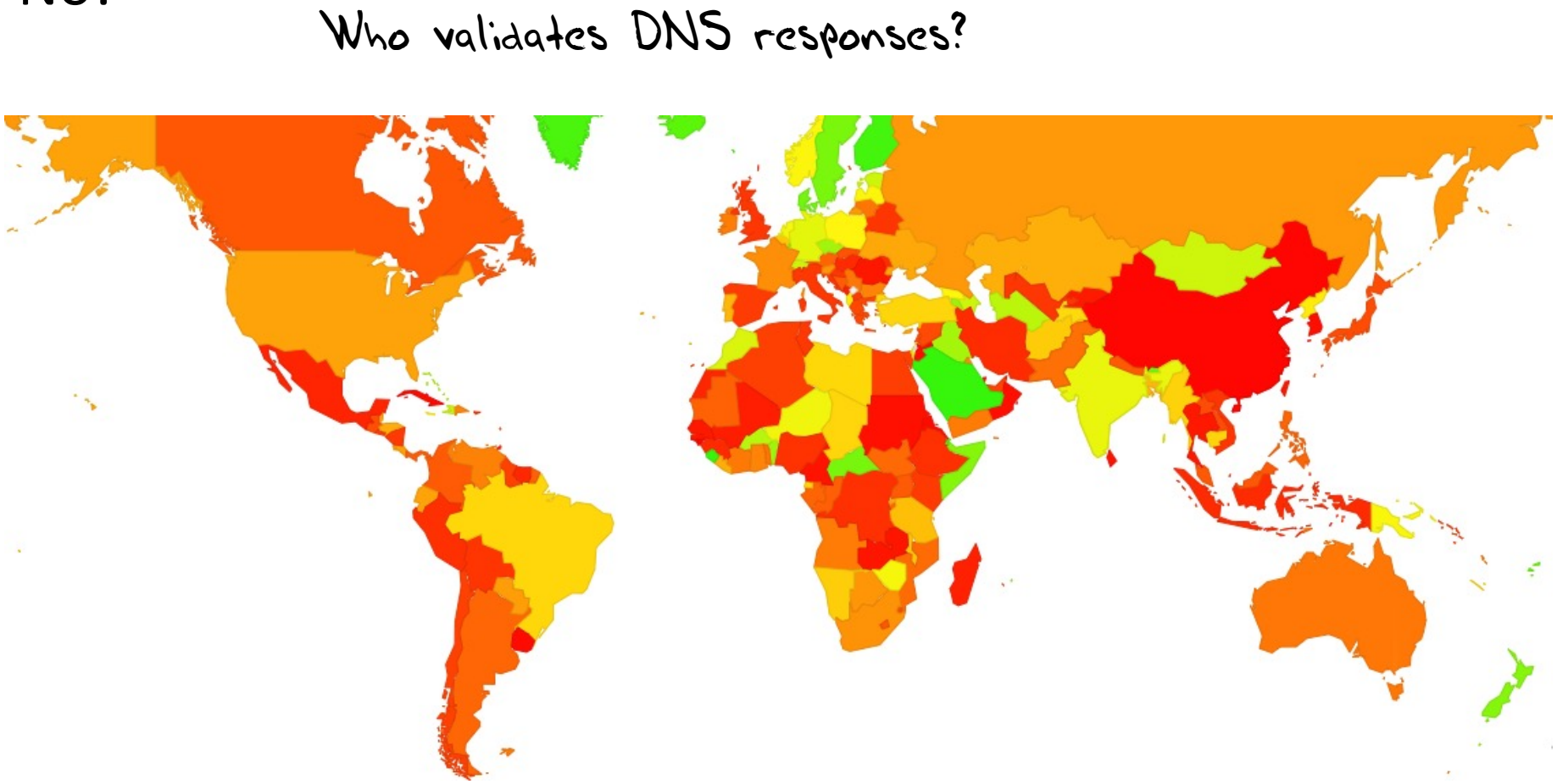
- Request a digital signature along with the DNS answer and validate the answer using a pre-provisioned trusted key (DNSSEC)

Is DNSSEC being used?

- Yes and No!

Is DNSSEC being used?

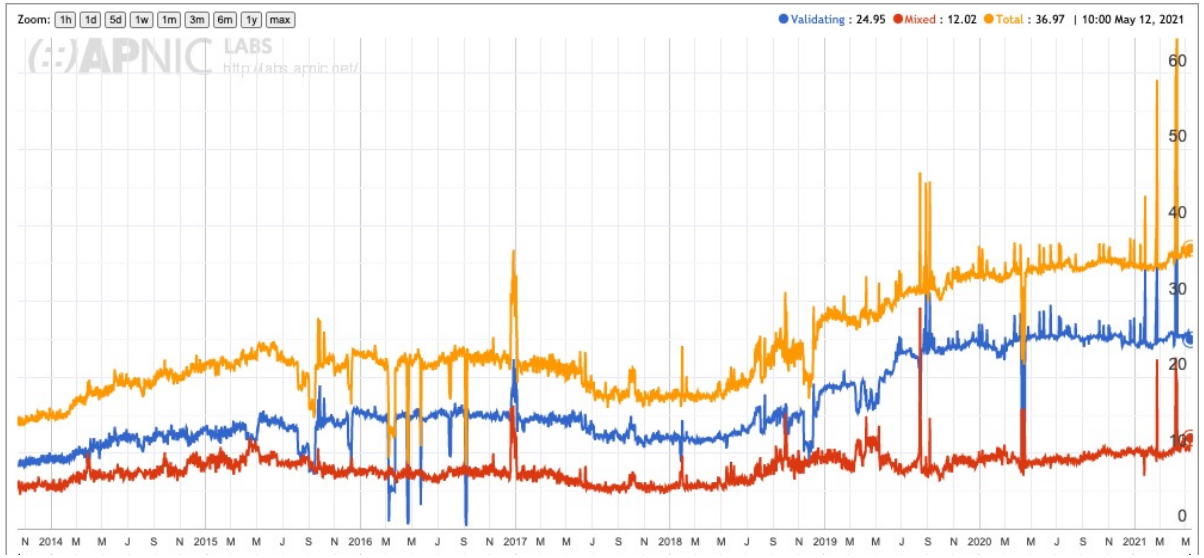
- Yes and No!



Is DNSSEC being used?

- Yes and No!

Who validates DNS responses?



25÷ of users are behind DNSSEC-validating resolvers who will not resolve a badly signed DNS name

Is DNSSEC being used?

- Yes and No!

Who signs DNS Zones?

?

Public data on the DNSSEC zone signing rate is hard to define, and even harder to come by!

Problems with DNSSEC

- Large DNS responses cause robustness issues for DNS
  - Getting large responses through the network has reliability issues with UDP packet fragmentation and timing issues with signalled cut-over to TCP
  - The validator has to perform a full backtrace query sequence to assemble the full DNSSEC signature chain
  - So the problem is that DNSSEC validation may entail a sequence of queries where each of the responses may require encounter UDP fragmentation packet loss

Some More Problems with DNSSEC

- Cryptographically “stronger” keys tend to be bigger keys over time, so the issue of cramming more data into DNS transactions is not going away!
- The stub-to-recursive hop is generally not using validation, so the user ends up trusting the validating recursive resolver in any case
- The current DNSSEC framework represents a lot of effort for only a marginal gain

Are we getting better at DNSSEC?

There is still a lot of room to improve our DNSSEC story

- Reducing validation-chain query delays using DNSSEC Chain responses?
- Using “denser” crypto algorithms to limit key and signature sizes?
- Using TCP for DNSSEC queries?
- NSEC3? Really?
- NSEC5? YMBK!

Authenticity in the DNS

- DNSSEC Validation cannot not prevent DNS eavesdropping, interception or tampering – all it can do is withhold DNS responses that are not “genuine”
- DNSSEC adoption is a trade-off in terms of additional costs of added points of fragility, added delay and load points balanced against the increased assurance of being able to place trust that the DNS responses are authentic

1. Adding authenticity to the DNS
2. **Increasing the reliance on the DNS for application level rendezvous functions**
3. Plugging DNS information leaks

It used to be so simple

- Query the DNS with a service name
- Get a response with the IP host address where the service is located
- Use the application to negotiate a service with the addressed host
- All services that share a common name share a common host

But we wanted more:

- We wanted to make a distinction between the service name and the platform that hosted the service
  - We wanted to have different services accessible using the same service name
  - We wanted a collection of platforms to deliver the service associated with a single service name
  - We wanted to outsource different services to different service providers
  - We wanted to steer the user to the “right” service provider for each user
  - And we wanted it to be FAST!

  - The concept of “go anywhere first and get redirected to an optimal service delivery point” is considered to be not FAST

So we added Bells and Whistles

- Put all of this optimisation into the DNS by:
  - Mapping the service names to host names
    - CNAME, DNAME and ANAME
    - None of these are very satisfactory!
  - The SRV record
    - This is either a swiss army knife or a chain saw massacre!
    - Add the service name (port) and protocol (transport) to the service name and use this as the query
    - And get the DNS response to come back with a collection of service delivery points
  - The Client Subnet query extension
    - Tag the query with the querier to permit tailoring of the service response in the DNS rather than in the application

More Bells (and Whistles!)

- SVCB and HTTPSSVC Resource Records
  - The “mega” response that can provide Application Level Protocols, IPv4 and IPv6 addresses, ESNI key, priority
  - Oh, and yes, there is an “alias form” that allows alias mapping at a zone apex

It's faster, but…

- But as we add more instrumentation to the DNS, it becomes a generic rendezvous tool, where a client forms a query based on an intended service access and the DNS response provides a set of service connection parameters that is potentially tailored to optimise the delivered service
- This means that real time knowledge of a user's DNS queries is synonymous to knowledge of a user's immediate intentions
- Which means that the DNS privacy issues become even more critical

1. Adding authenticity to the DNS
2. Increasing the reliance on the DNS for application level rendezvous functions
3. **Plugging DNS information leaks**

Plugging the DNS leakage

- **Query Name Minimisation** to reduce the extravagant chattiness of the DNS resolution process on the recursive to authoritative path
- **DNS over TLS** on the stub to recursive path
  - Channel protection, remote end authentication and transport robustness
- **DNS over HTTPS (/3)** on the stub to recursive path
  - Channel protection, remote end authentication, transport robustness and HTTP object semantics
- **Oblivious DNS over HTTPS (/3)** on the stub to recursive path
  - Hide the implicit end point identity / query name association leakage

Coming soon?

- Extending DNS channel protection to the recursive to authoritative hops

  (Although this may be tougher than it looks at first!)

Scaling with Encrypted Channels

- Session level encryption involves session establishment and maintenance overhead
  - Typically this entails a TCP overhead (direction or within a QUIC envelope) and a TGLS overhead
  - This can be amortised through session reuse
  - Session reuse is most effective on the stub to recursive paths
- The secure Web infrastructure points to ways that we can scale an encrypted DNS infrastructure, but the economics of the DNS are somewhat different than those of the web

Will all this be deployed?

Can we do this?

- Pretty clearly we have most of the tools available to achieve all of these objectives
  - Leverage TLS to provide session level encryption
  - Leverage HTTPS to push stub resolution functions into applications
  - Use the DNS HTTPSSVC to provide the ESNI key
- Yes we can!

Will we do this?

- This is a far more challenging question!

If HTTPS worked, why not DoH?

- Any change to the DNS that requires user configuration, or a change of CPE behaviour will not be easy to gather deployment momentum
- There is no untapped financial return in DNS resolution, so this is not an activity that has strong commercial impetus
- Many public environments use DNS oversight and alteration as a means of content moderation. There is little appetite to make that harder
- Browser vendors have far more limited leverage in the DNS, as compared to content delivery over HTTP

The DNS Economy

- In the public Internet, end clients don't normally pay directly for DNS recursive resolution services
- Which implies that outside of the domain of the local ISP, DNS resolvers are essentially unfunded by the resolver's clients
- And efforts to monetise the DNS with various forms of funded misdirection (such as NXDOMAIN substitution) are generally viewed with extreme disfavour
- Open Resolver efforts run the risk of success-disaster
  - The more they are used, the greater the funding problem to run them at scale
  - The greater the funding problem the greater the temptation to monetise the DNS resolver function in more subtle ways

The DNS Economy

- The default option is that the ISP funds and operate the recursive DNS service, funded by the ISP's client base
  - 70% of all end clients use same-AS recursive resolvers *
- However the fact that it works today does not mean that you can double the input costs and expect it to just keep on working tomorrow
- For ISPs the DNS is a cost department, not a revenue source
  - We should expect strong resistance from ISPs to increase their costs in DNS service provision
- The DNS is also highly resistant to changes in the edge infrastructure

* https://stats.labs.apnic.net/rvrs

Where is this heading?

- Will any of these privacy approaches becomes mainstream in the public Internet?

My Opinion

- ISP-based provisioning of DNS servers without channel encryption will continue to be the mainstream of the public DNS infrastructure
- Most users don’t change their platform settings from the defaults and CPE based service provisioning in the wired networks and direct provisioning in mobile networks will persist

My Opinion

- ISP-based provisioning of DNS servers without channel encryption will continue to be the mainstream of the public DNS infrastructure
- Most users don’t change their platform settings from the defaults and CPE based service provisioning in the wired networks and direct provisioning in mobile networks will persist
- But that’s not the full story...

"Split" DNS

- Is appears more likely that applications who want to tailor their DNS use to adopt a more private profile will hive off to use DoH to an application-selected DNS service, while the platform itself will continue to use libraries that will default to DNS over UDP to the ISP-provided recursive DNS resolver
- That way the application ecosystem can fund its own DNS privacy infrastructure and avoid waiting for everyone else to make the necessary infrastructure and service investments before they can adopt DNS privacy themselves
- The prospect of **application-specific naming services** is a very real prospect in such a scenario

It's life Jim, but not as we know it!*

- The overall progression here is an evolution from network-centric services to platform-centric services to today's world of application-centric services
- It's clear that the DNS is being swept up in this shift, and the DNS is changing in almost every respect
- The future prospects of a single unified coherent name space as embodied in the DNS, as we currently know it, for the entire internet service domain are looking pretty poor right now!

* With apologies to the Trekies!

Thanks!