DNS Privacy

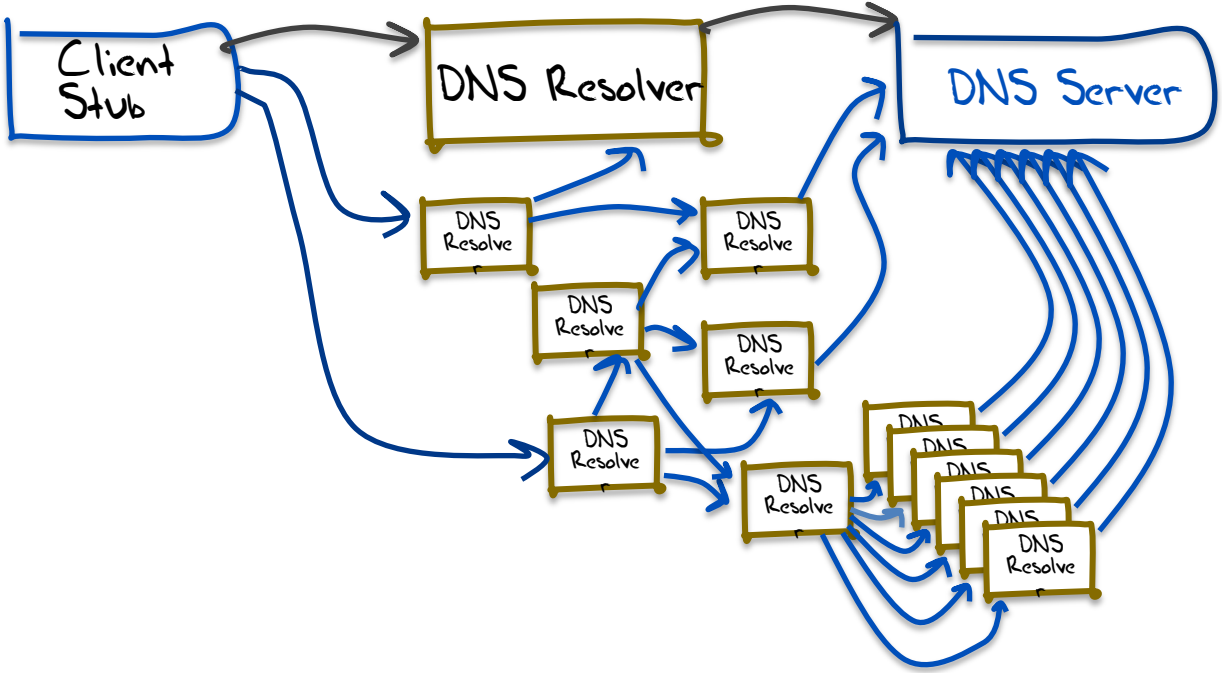
Geoff Huston AM
APNIC Labs

DNS Surveillance

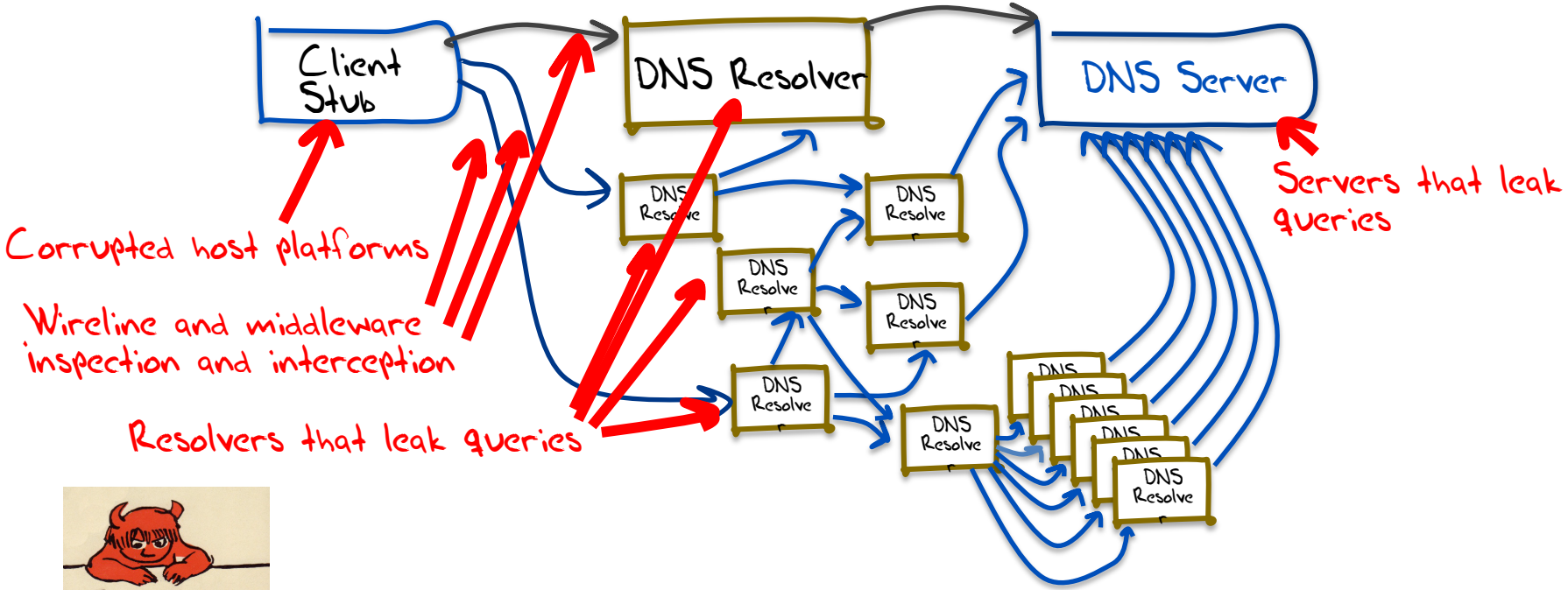
- The DNS is used by many actors as a means of looking at what we do online and censoring what services we can access online
- Can we stop DNS surveillance completely?
  - Probably not!
- Can we make it harder for others to collect individual profiles of activity?
  - Well, yes, we can!
  - And that's what I want to talk about today

How we might think the DNS works

What we suspect the DNS is like



What we suspect the DNS is like



Why pick on the DNS?

- The DNS is very **easy to tap**
  - Its open and unencrypted
- DNS traffic is **easy to tamper with**
  - Its payload is not secured and tampering cannot be detected
  - Its predictable and false answers can be readily inserted
- The DNS is **hard to trace**
  - Noone knows exactly where their queries go
  - Noone can know precisely where their answers come from

Second-hand DNS queries are a business opportunity these days

How can we improve DNS Privacy?

- Lets look at a few behaviours of the DNS and see what we are doing to try and improve its privacy properties

I. The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve www.example.com

Not me – try asking the servers for example.com

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve www.example.com

Not me – try asking the servers for example.com

Hi example.com server, I want to resolve www.example.com

Sure – its 93.184.216.34

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Why are we telling root servers all our DNS secrets?

In our example case, both a root server and a .com server now know that I am attempting to resolve the name www.example.com

Maybe I don't want them to know this

The DNS is overly chatty

Is there an alternative approach to name server discovery that strips the query name in iterative search for a zone's servers?

Yes – the extra information was inserted into the query to make the protocol simpler and slightly more efficient in some cases

But we can alter query behaviour to only expose as much as is necessary to the folk who need to know in order to answer the query

QNAME Minimisation

- A resolver technique intended to improve DNS privacy where a DNS resolver no longer sends the entire original query name to the upstream name server
- Described in RFC 7816

> Instead of sending the full QNAME and the original QTYPE upstream, a resolver that implements QNAME minimisation and does not already have the answer in its cache sends a request to the name server authoritative for the closest known ancestor of the original QNAME. The request is done with:
>
> o the QTYPE NS
>
> o the QNAME that is the original QNAME, stripped to just one label more than the zone for which the server is authoritative

Example of QNAME Minimisation

Ask the authoritative server for a zone for the NS records of the next zone:

Hi Root server, I want to know the nameservers for com

Sure, here are the servers for .com

Example of QNAME Minimisation

Ask the authoritative server for a zone for the NS records of the next zone:

Hi Root server, I want to know the nameservers for com

Sure, here are the servers for .com

Hi .com server, I want to know the nameservers for example.com

Sure, here are the servers for example.com

Example of QNAME Minimisation

Ask the authoritative server for a zone for the NS records of the next zone:

Hi Root server, I want to know the nameservers for com

Sure, here are the servers for .com

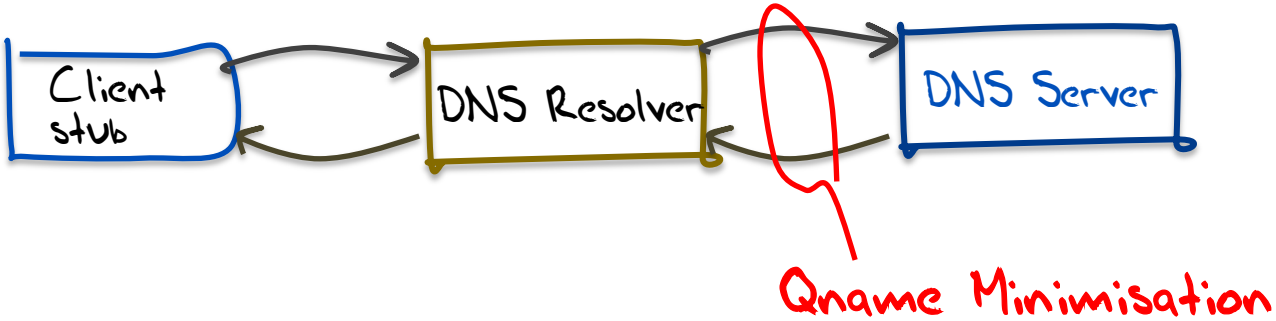
Hi .com server, I want to know the nameservers for example.com

Sure, here are the servers for example.com

Hi example.com server, I want to resolve www.example.com

Sure – its 93.184.216.34

DNS Privacy



II. Interception and Rewriting

- The DNS is an easy target for the imposition of control over access
  - Try asking for www.thepiratebay.org in Australia
  - Try asking for www.facebook.com in China
  - And on and on and on…
- These days interception systems typically offer an incorrect response
- **How can you tell is the answer that the DNS gives you is the genuine answer or not?**

DNSSEC

- DNSSEC adds additional information into the DNS
  - a digital signature record is added to all RRsets in a zone, signed by the zone controller
  - Any third party who tries to alter a signed zone is unable to generate an authentic signature (as they do not know the zone key value)
- DNSSEC validation of the signed DNS response can tell you if the response is genuine or if it is out of date or has been altered
  - DNSSEC can’t tell you what the “good” answer is, just that the answer you got was not it!
- DNSSEC will also tell if is an NXDOMAIN response is authentic

DNSSEC and Recursive Resolvers

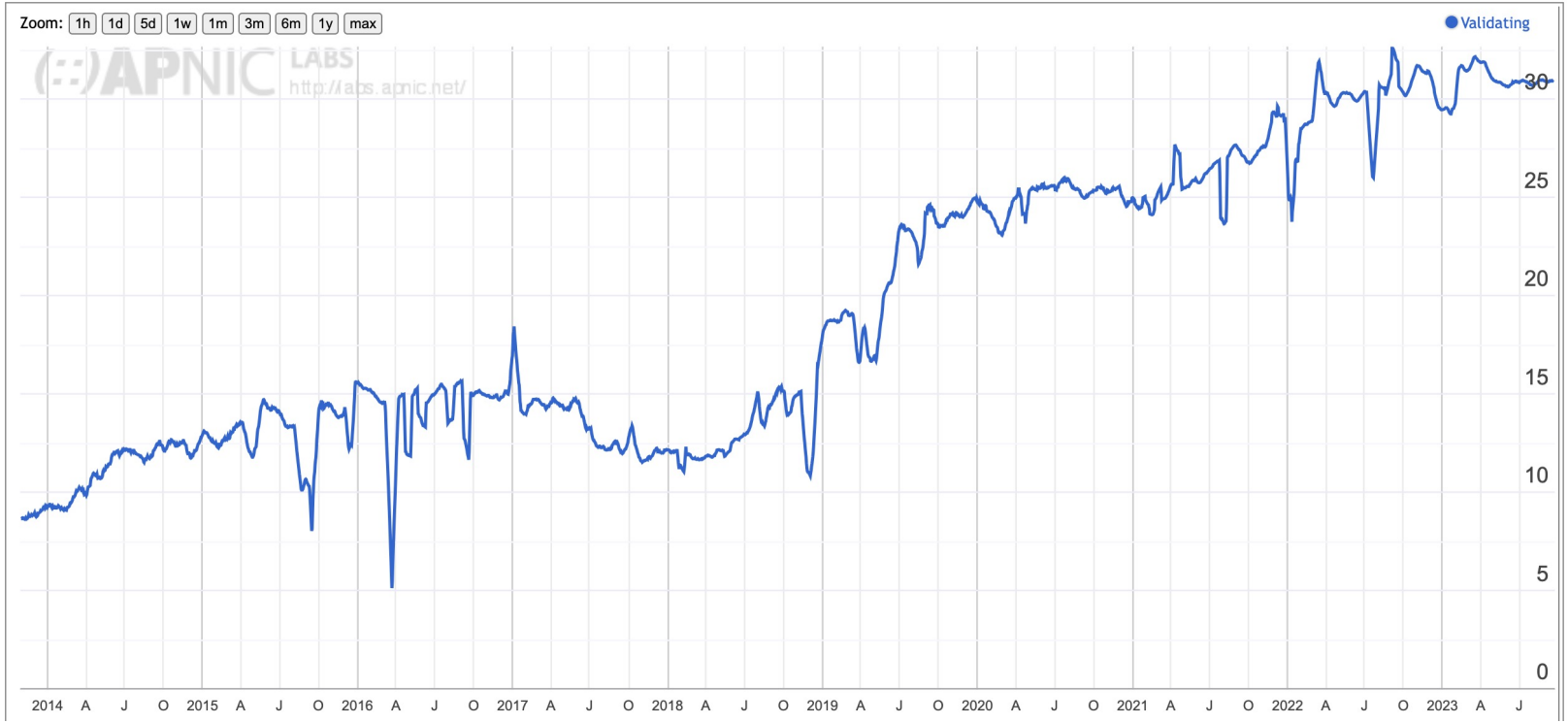
- A DNS response that has been modified will fail to validate.

  When:

  - a client asks a security-aware resolver to resolve a name, and
  - sets the EDNS(0) DNSSEC OK bit, and
  - the zone is DNSSEC-signed

  then the recursive resolver will only return a RRset for the query if it can validate the response using the attached digital signature

DNSSEC Validation Use



30%

DNS Privacy


Client stub
DNS Resolver
DNS Server
Qname Minimisation
DNSSEC Validation

III. Wire Tapping and Inspection

- The DNS is an open (unencrypted) protocol
- If we want to stop third party inspection we need to encrypt the transport used by DNS queries and responses
  - Today the standard tool is TLS, which uses dynamically generated session keys to encrypt all traffic between two parties
  - We could use TLS between the end client and the client's recursive resolver

DNS over DTLS

- DTLS is a UDP variant of TLS that is intended to work over UDP rather than TCP (RFC 8094)
- However:
  - DTLS is intolerant of fragmentation
  - It appears to have similar overheads to TLS
  - I’m not sure if there are any robust implementations of DNS over DTLS

DNS over TLS (DoT)

- TLS is a TCP ‘overlay’ that adds server authentication and session encryption to TCP
- TLS uses an initial handshake to allow a client to:
  - Validate the identity of the server
  - Negotiate a session key to be used in all subsequent packets in the TCP session
- Best used between the stub resolver and its recursive resolver in persistent session mode

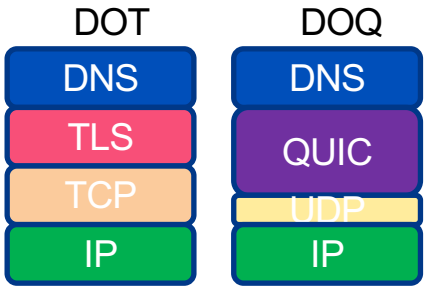
DNS over TLS (DoT)

- The queries and the responses are hidden from intermediaries – preventing wiretapping from revealing DNS queries and responses
- The client can validates the recursive resolver's identity – preventing third parties from intercepting DNS queries and generating fake responses

DNS over TLS (DoT)

- Will generate a higher recursive resolver load as stub client may have a held state with the recursive resolver
- The TCP session state is on port 853
  - DNS over TLS can be readily blocked by middleware
- The privacy is relative, as the recursive resolver still knows all your DNS queries

DNS over QUIC (DoQ)



- QUIC is a transport protocol originally developed by Google and passed over to the IETF for standardised profile development
- QUIC uses a thin UDP shim and an encrypted payload
  - The payload is divided into a TCP-like transport header and a payload
- The essential difference between DOT and DOQ is the deliberate hiding of the transport protocol from network middleware with the use of QUIC

DNS over QUIC DoQ

- Has much the same benefits as DNS over TLS, but adds the ability to have multiple outstanding queries between the stub and the recursive

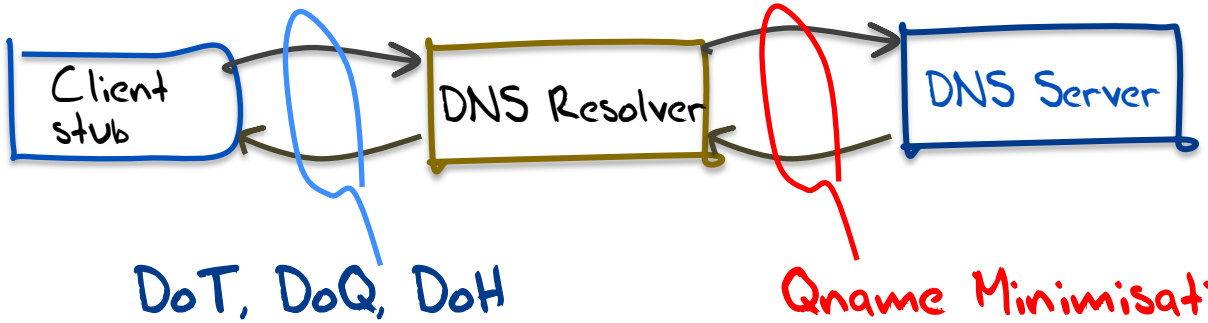
DNS over HTTPS/2 (DoH)

- Uses an HTTPS session with a resolver
- Similar to DNS over TLS, but with HTTP object semantics
- Uses TCP port 443, so can be masked within other HTTPS traffic
- Can use DNS wireformat or JSON format DNS payload

DNS over HTTPS/3 (DoH)

- Uses an HTTPS session with a resolver
- Similar to DNS over QUIC, but with HTTP object semantics
- Uses UDP port 443, so can be masked within other HTTPS/3 traffic
- Can use DNS wireformat or JSON format DNS payload

DNS Privacy


Client stub
DNS Resolver
DNS Server
DoT, DoQ, DoH
Qname Minimisation
DNSSEC Validation

DNS within the Browser

- Firefox's "Trusted Recursive Resolver"
- Avoids using the local DNS resolver library and local DNS infrastructure
- Has the browser sending its DNS queries directly to a trusted resolver over HTTPS
- Servers available from Cloudflare, Google, Cleanbrowsing

Choose your resolver carefully

- The careful choice of an open recursive resolver and an encrypted DNS session will go a long way along the path of DNS privacy
- But the compromise is that you are sharing your activity profile with the recursive resolver operator
- But this need not be the case

DoH and Push DNS

- HTTPS allows server push objects as well as pull/get methods
- This can allow a web page to push a DNS result to the client without the client performing any DNS query at all
- When used with DNSSEC and Chained Validation responses the client can ensure that the DNS data is valid and current
- Its fast and very private

Obfusified DNS

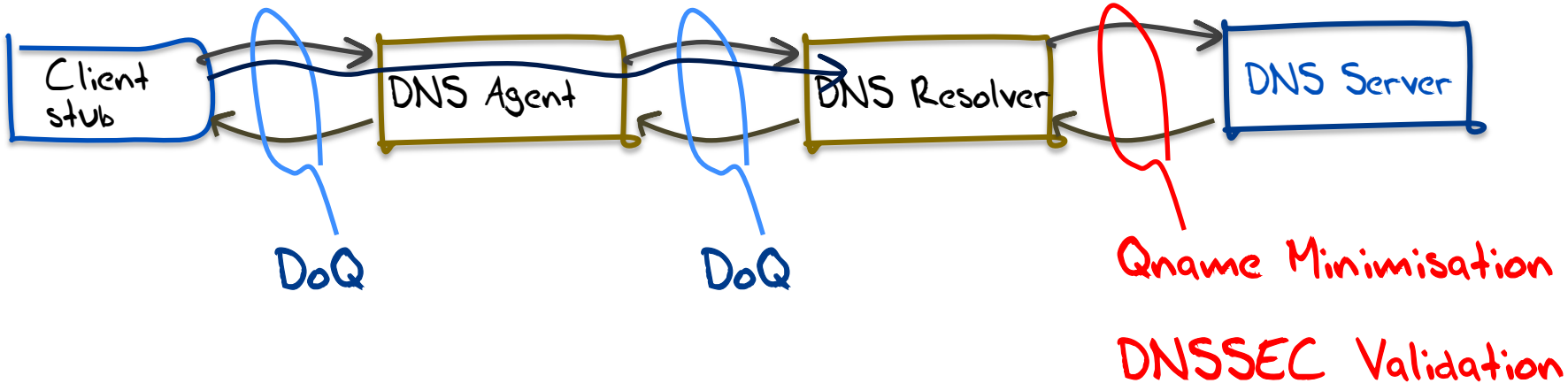
- Use two separate agents in place of a single recursive resolver
- Use double encryption wrapping
- The first agent “knows” the IP identity of the client, but not the DNS query that the client has generated
- The second agent “knows” the query, but not the IP identity of the client

Obfusified DNS

- This approach can extremely effective in preserving end user privacy in the DNS
- This approach has been used in Apple's Private Data Relay service as a way of protecting the user from third party observervation

DNS Privacy

DNS Obfusication



EDNS Client Subnet

- There is a tension between CDN operators that rely on customized DNS responses to perform content steering, and the use of large scale open resolvers that do not necessarily preserve locality
  - The CDN wants to use the assumed location of the DNS resolver to infer the location of the client and direct them to a nearby service instance
  - The result is that the CDN operators want the stub client's IP subnet embedded in the query to ensure that the CDN could provide enhanced content steering for the client by geolocating this subnet

EDNS Client Subnet

- This has raised a number of concerns about DNS privacy
- There is a forming consensus that Client Subnet has been a step too far in terms of potential privacy leakage into the DNS

Privacy Tensions in the DNS

- Exposing the end user's IP location to the DNS leads to better outcomes for content server steering, but compromises user privacy
- Hiding the end user completely (DoH Push, Obfusified DNS) can lead to pretty comprehensive levels of privacy, and faster DNS operations, but can disrupt content server steering
  - Given that many CDN's are now reliant on DNS-based geolocation, there is current work to use an obscured subnet token value that geo-locates to a similar location, but is unrelated to the end client's IP address / subnet
  - Presumably, we could replace this subnet with a standard geolocator reference, were one defined as an industry standard

Thanks!