DNS Privacy (an update)

Geoff Huston
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 detected
- DNS responses are predictable and false answers can be readily inserted

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

DNS Surveillance

- Can we stop DNS surveillance completely?
  - Probably not!

- Can we make it harder to collect individual profiles of activity?
  - Well, yes
  - And that's what I want to talk about today

What's the problem here?

- Is the **DNS label** being queried a secret?
  - Well, not normally *
    - Although there are DNS versions of steganography that can conceal data in the query string

What's the problem here?

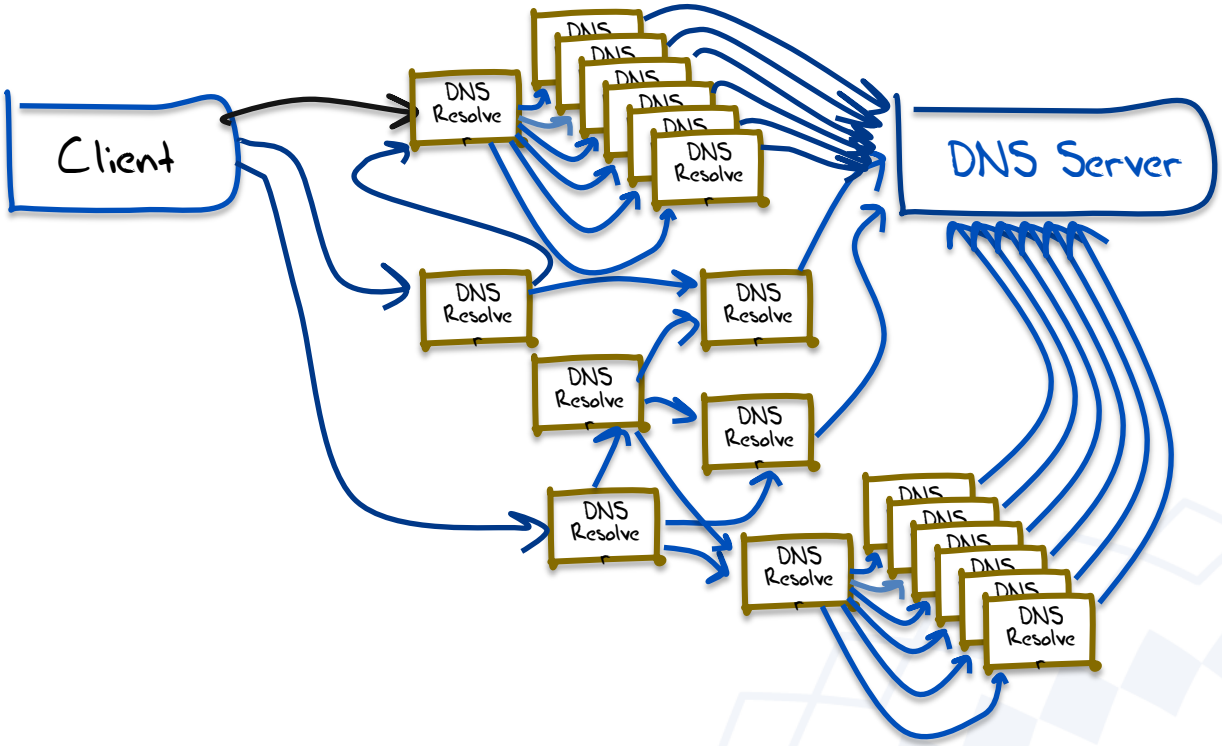
- Is the **DNS label** being queried a secret?
  - Well, not normally
- Is the **DNS response** to a query a secret?
  - Again, not normally *
    - Although there are DNS versions of steganography that can conceal data in the response value

What's the problem here?

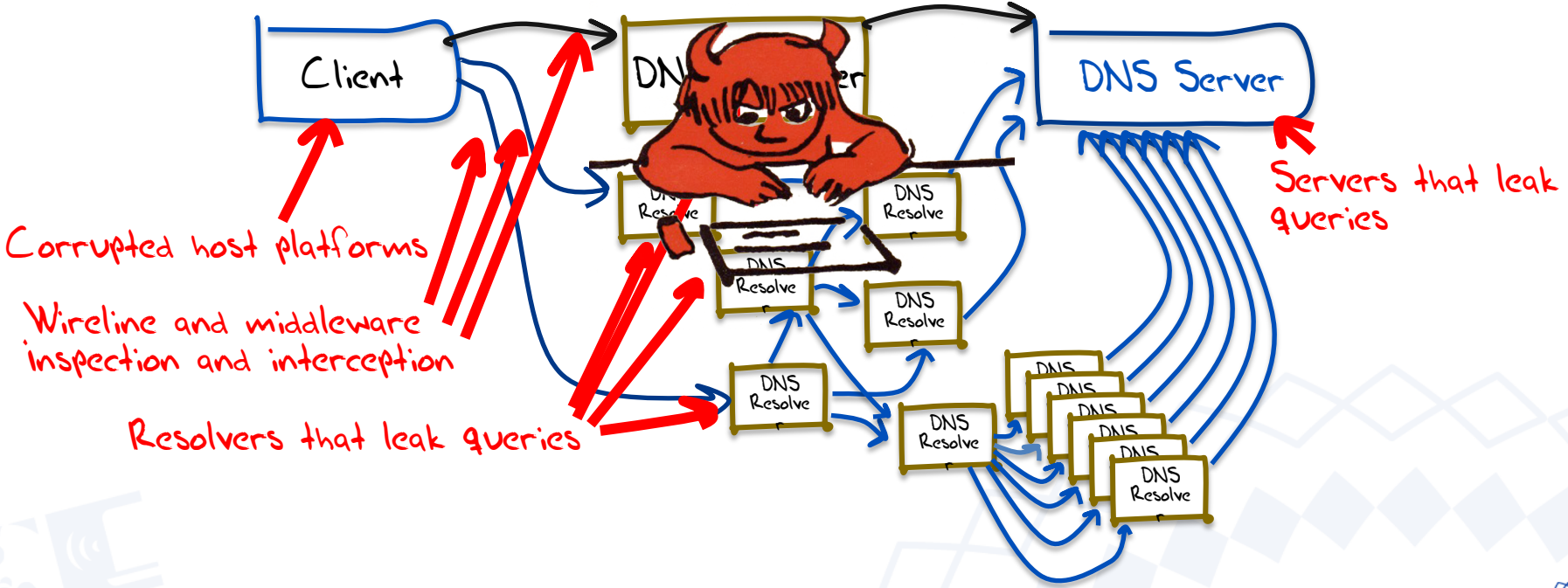
- Is the **DNS label** being queried a secret?
  - Well, not normally
- Is the **DNS response** to a query a secret?
  - Again, not normally
- So what is the issue here?
  - It’s the combination of query and the meta-data around a query that creates a problem:
    - The end user identity, from the IP packet header
    - The DNS label (or sequence of labels) being queried, from the payload
    - The date and time

How we might think the DNS works

What we suspect the DNS is like



What we suspect the DNS is like



Second-hand DNS queries are a business opportunity these days

How can we improve DNS Privacy?

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

1. 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 the A record for www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve the A record for www.example.com

Not me – try asking the servers for example.com

Hi example.com server, I want to resolve the A record for www.example.com

Sure – its 93.184.216.34

The DNS is 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

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

Yes, but…

It’s a technique to minimise the information leak between a recursive resolver and authoritative servers, as stub resolvers pass the full query label to the recursive resolver

A number of commonly used recursive resolvers only perform qname minimisation on the first three labels

(Why do they limit Qname minimisation to just the upper level domains)?

2. 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

  Etc, etc
- These days interception systems typically offer an **incorrect response**
- How can you tell if the answer that the DNS gives you is the genuine answer or not?
  - This sounds like a question for DNSSEC!

DNSSEC and Recursive Resolvers

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

- A DNSSEC-validating recursive resolver will only return a RRset for the query if it can validate the response using the associated digital signature, and It will set the AD bit in the resolver response to indicate validation success Otherwise it will return SERVFAIL

- But SERVFAIL is not the same as “I smell tampering”
  - Its ”nope, I failed. Try another resolver”

Yes, but…

- The zone (and all its parent zones) must be DNSSEC-signed
- If the recursive resolver performs DNSSEC validation (using the recursive resolver to validate is the most prevalent deployment model) then the all-important stub-to-recursive link is still vulnerable to interception and re-writing
- And if your recursive resolver is performing the re-writing of the response then the stub is none the wiser if the stub does not perform DNSSEC validation
- Stub resolvers don't generally perform DNSSEC validation
  - It's too slow!

3. Middleware and WireTapping

- If we want to make DNS surveillance harder we should look at encrypting the transport used by DNS queries and responses between stub and recursive resolvers
- Today's standard tool is TLS, which uses dynamically generated session keys to encrypt all traffic between two parties

DoT - DNS over TLS

- TLS is a TCP ‘overlay’ that adds server authentication and session encryption to TCP
- DoT uses a persistent stub-to-recursive relationship to amortize the setup costs of TCP and TLS over many subsequent queries
  - Which works efficiently in a stub-to-recursive scenario, but not even a little bit for recursive-to-authoritatives!
- If the initial server name certificate is validated by the client then
  - The client can be assured (to some extent) of who it is talking to by name
  - No third parties can intrude or observe the DoT session or its contents

Yes, but…

- The TCP session state is on port 853
  - DNS over TLS can be readily blocked by CPE and middleware
- Will generate a higher recursive resolver memory load as each client may have a held state with one or more recursive resolvers
- The privacy is relative, as the recursive resolver still knows all about you and your DNS queries
- And until ECH* in TLS 1.3 is widely supported, the identity of the TLS server is still in the clear, which also facilitates blocking even if the DoT session jumps over to use TCP port 443

* Encrypting the Server Name in the Client Hello message of TLS setup

DoH - DNS over HTTPS

- DNS over HTTPS
- Uses an HTTPS session for the stub-to-recursive link
- Similar to DNS over TLS, but with HTTP object semantics interposed between the DNS and TLS
- Uses TCP port 443, so can be masked within other HTTPS traffic
- Uses DNS wire format

Why use DoH over DoT?

- Bypass middleware blocking of TCP port 853 (DoT)

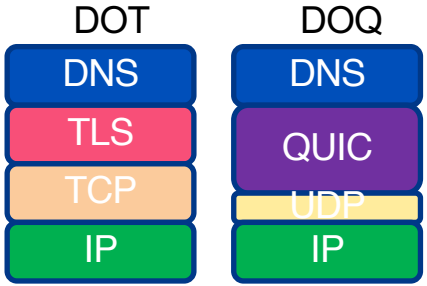
- DoH allows the stub resolver function to be merged into the application at the client end and DNS resolver to be multiplexed at the server side (browsers and web servers)
  - HTTP object semantics allow for HTTP object caching in the client
  - This enables server-side HTTP push of DNS responses
  - Resolver-less DNS!
  - Can speed up transactions through pre-provisioning of DNS responses

Yes, but…

- Aside from changing the TCP port to 443 there is little difference between DoH and DoT from a conventional DNS perspective
- Most of the issues with DoH are about the use of resolver-less DNS and content-based DoH-server switching using the HTTP framing shim, which are still largely speculative matters these days
- Application-level DoH can be readily hidden from the platform and from the local network – this can be seen as a good or bad thing!

DNS over QUIC

- 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
- QUIC allows for multiple DNS queries without TCP HOL blocking



Yes, but…

- QUIC on UDP port 443 has issues with port blocking in middleware
- There is little difference between DoQ, DoH and DoT from a conventional DNS perspective
  - The remote end recursive resolver still is privy to all your DNS queries and your identity

DoH again!

- DNS over HTTPS/3
  - From the perspective of the DNS its still DNS binary objects encased in an HTTP wrapper, using POST for the query and a HTTP Data Frame for the response
  - From the perspective of the network, HTTP/3 can negotiate the use of QUIC as its transport network
- DoH is morphing into DNS-over-HTTPS-over-QUIC-over-UDP at about the same speed as HTTP/2 is morphing into HTTP/3 that incorporates TLS 1.3 into QUIC

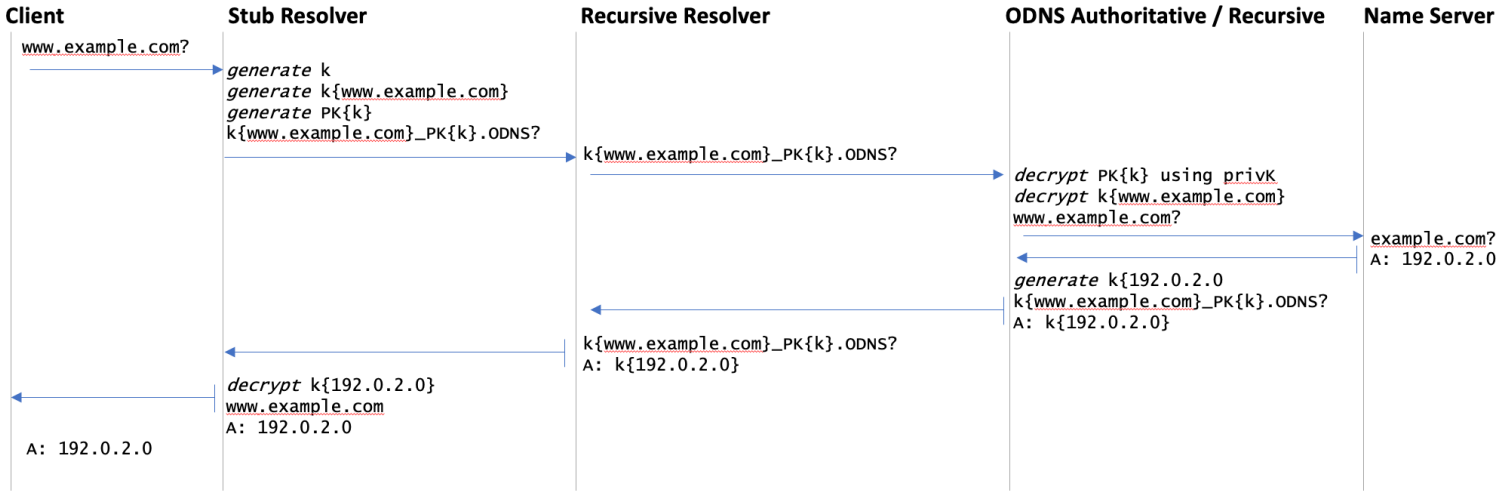
4. DNS Profiling

- Your identity and the sequences of your DNS queries represent a rich vein of profiling information
- Can we deconstruct the DNS in such a way that no single party has both pieces of information?

Oblivious DNS

- Uses the QUERY name to disassociate stub identity from query
  - Stub resolver encrypts the DNS query label into a new query label
    - Encryption uses the public key of a known oDNS server, and appends the name of the oDNS server
  - Stub resolver queries a ‘normal’ recursive resolver with this encrypted query name
  - Recursive resolver queries an oDNS server with this encrypted query name
  - oDNS server strips out its own name and decrypts the query name, and resolves this name and encrypts the DNS RR to send back to the stub via the recursive resolver

Oblivious DNS

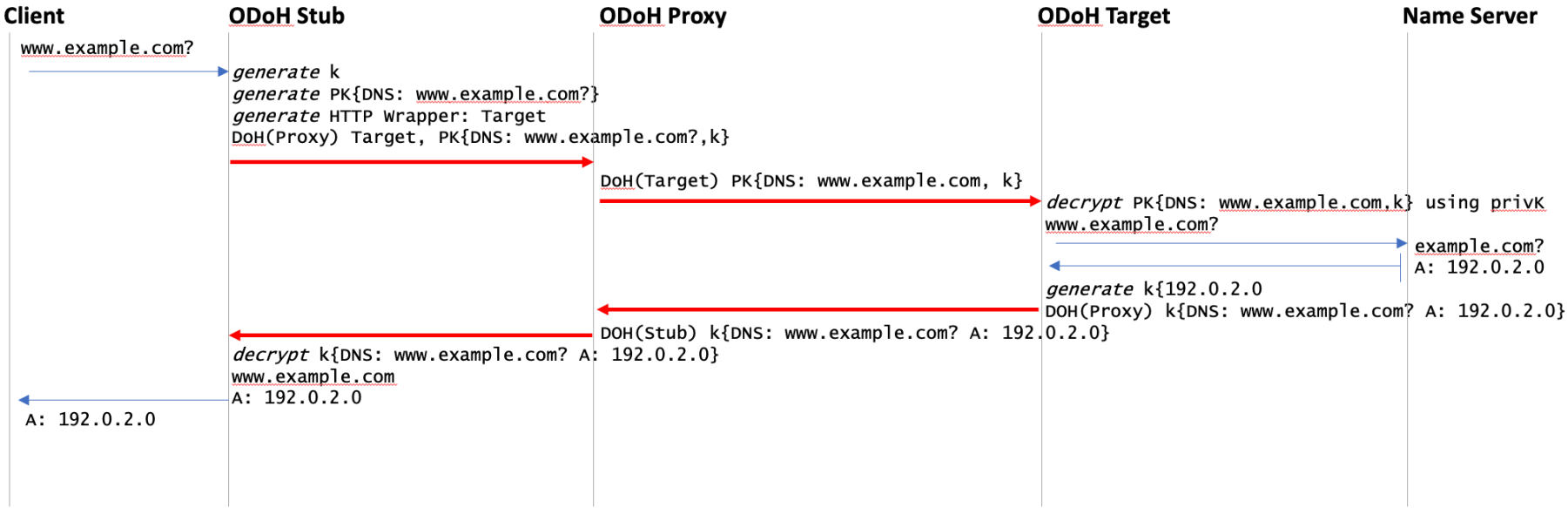


Yes, but…

- The DNS is still DNS over UDP port 53
  - But nothing prevents a oDNS stub using Do[THQ] to a recursive resolver. The recursive resolver has no knowledge of oDNS and process the DNS query like any other
- The encryption is limited due to limited size and alphabet of the query name field

Oblivious DoH

- Use double TLS wrapper on a DoH transport to dissociate query name from stub identity



Oblivious DoH

- An outer TLS wrapper is used for the stub-to-oDoH Proxy hop and a different TLS wrapper is used for the oDoH Proxy-to-oDoH Target hop
- The inner TLS wrapper is used to encrypt the DNS query, encrypted using the public key of the target
- The response is encrypted using a session key generated by the client

Yes, but…

- This requires a modified DNS stub resolver that can send and receive oDoH messages, an ODoH Proxy and an ODoH Target
- Oh, and the ODoH proxy and the ODoH Target should not collude!
  - But we can’t ensure that no collusion happens!

Obscured DNS

- Borrowed from the approach used by IDNS
- Apply a hash the zone file by passing the zone labels through a hash to get a base32hex encoded version of the labels, keyed with a passphrase
- The encrypted zone is published through conventional DNS
- Qnames need to be encrypted before passing them to the DNS
- Only holders of the common passphrase can decrypt the responses – no DNS intermediary can determine the original query label

Yes, but…

- “shared secrets” are often an oxymoron!

Hiding in the Crowd

- What if you use an encrypted session to a very busy open resolver?
  - No third party can see you queries to the open resolver
  - Noone else can see the responses from the open resolver
  - The open resolver asks the authoritative servers which makes it challenging to map the query back to the end user

  - So if you you are prepared to trust Google, Open DNS, Cloudflare, Quad9, etc with your DNS, and you use DoH or DoT on the stub-to-recursive hop then its far harder for any third party to associate your identity with your queries
    - But that is a very large amount of trust you are investing here in folk whom you are not paying to provide this service!

Hiding in the Crowds

- What if you round-robin your queries to a number of open resolvers?
  - No single open resolver provider can see your complete DNS query set
  - Which makes profiling at the open resolver more challenging
    - Even though many open DNS providers assert that they do not retain queries nor profile users in any case

Where is this heading?

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

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 local ISP, DNS resolvers are essentially unfunded by their clients
  - And efforts to monetise the DNS (such as NXDOMAIN substitution) are generally viewed with disfavour
  - Open Resolver efforts run the risk of success-disaster
    - They more they are used the greater the funding problem
- The default option is that the ISP funds and operate the recursive DNS service, funded by the ISP's client base

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 from the defaults and CPE based service provisioning in the wired networks and direct provisioning in mobile networks will persist for the moment
- Some applications will shift to DoH support, but on the whole will continue to use the default ISP-based resolvers (assuming that they include DoH support)

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

"Split" DNS

- Is appears likely that browsers will hive off to use DoH to the ISP default recursive resolver, while the platform itself will continue to use libraries that will default to DNS over UDP
- Which will produce some awesome corner cases when failure modes are encountered!

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
- Or you could just take the default option, do nothing and pass your queries, along with your traffic, to your ISP!

Thanks!


2021
APRICOT
APNIC 51
ONLINE
22 February – 4 March 2021
#apricot2021