DNS Privacy (or not!)

Geoff Huston
APNIC

Street Art: Banksy

https://xkcd.com/1361/

Why?

- Because everything you do on the net starts with a call to the DNS
- If someone could see your stream of queries in real time then they could assemble a detailed profile of you and interests and activities
- Do we have any evidence of DNS data mining?
  - Data miners don’t disclose their sources as a rule
- How about something related:
  - Do we have any evidence of DNS stalking?

What if…

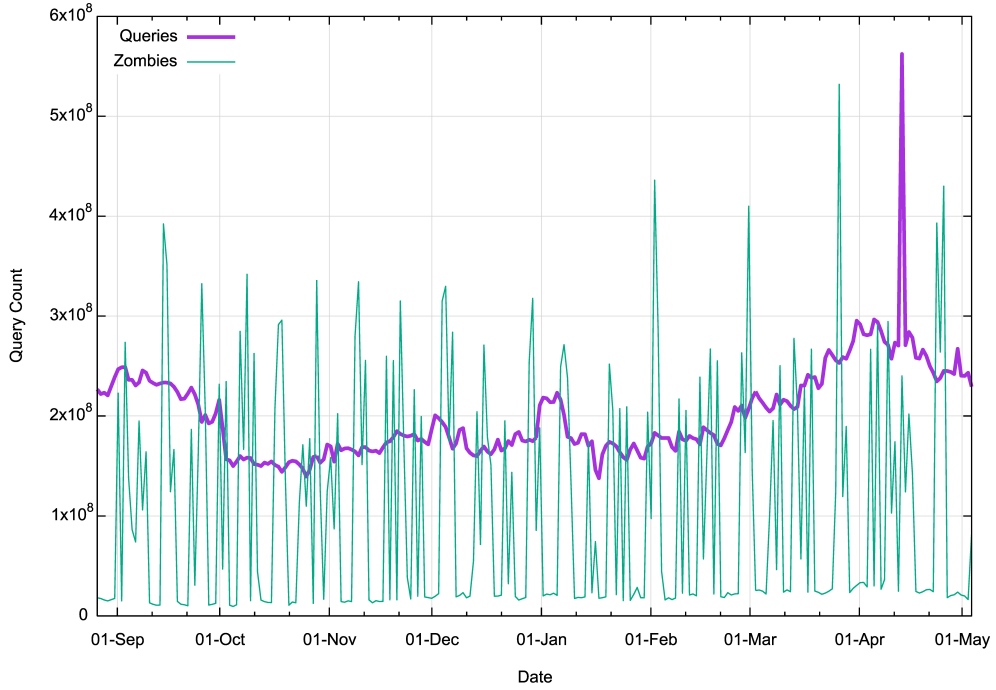
- I gave you an absolutely unique DNS name to resolve:
  - The name never existed before now
  - The name will never be used again
  - The name includes the time when the name was created
- If I am the authoritative server for the name's zone then I should see your efforts to resolve the name
- Then I should never see the name as a resolution query ever again

  Unless you have attracted a digital stalker who performs re-queries of your DNS names!

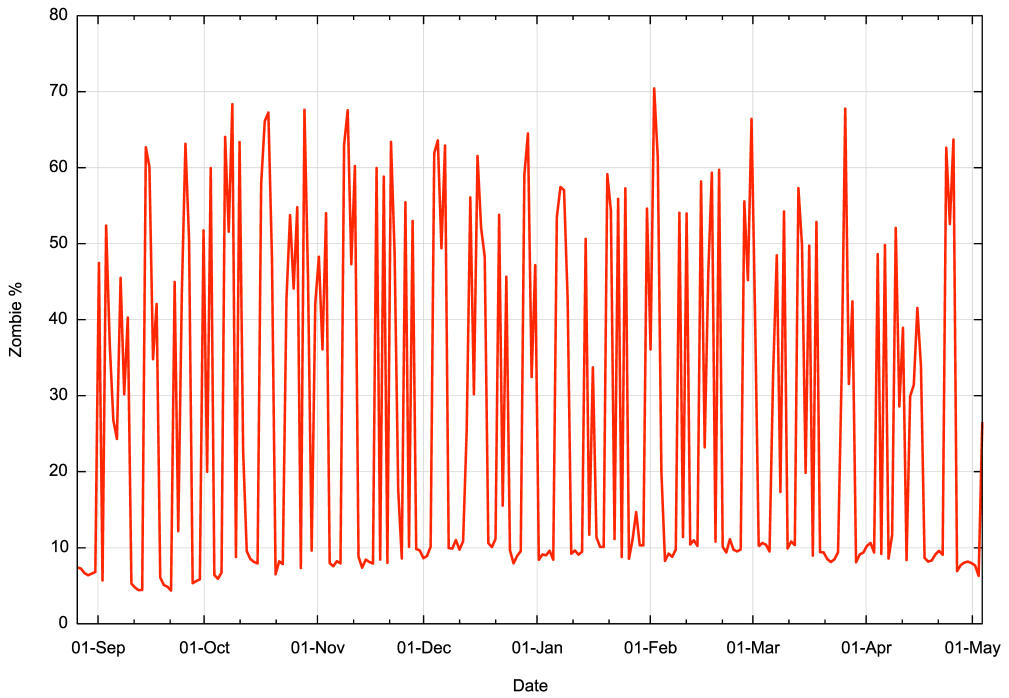
DNS Re-query Rate

- Over the past 9 months, a minimum of 6% - 8% of daily query totals are zombie queries, asking the same query more than 30 seconds after the initial query – and some days its as high as 66%

Daily Query Counts

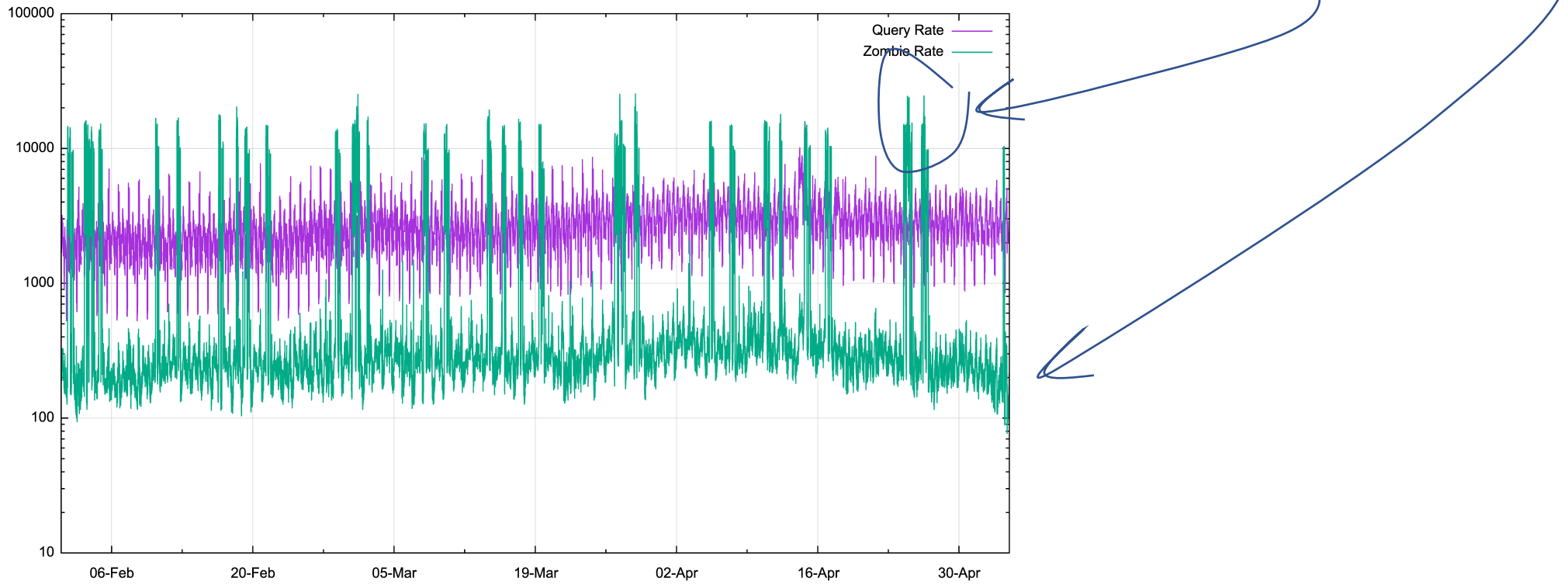


Daily Query to Zombie Ratios



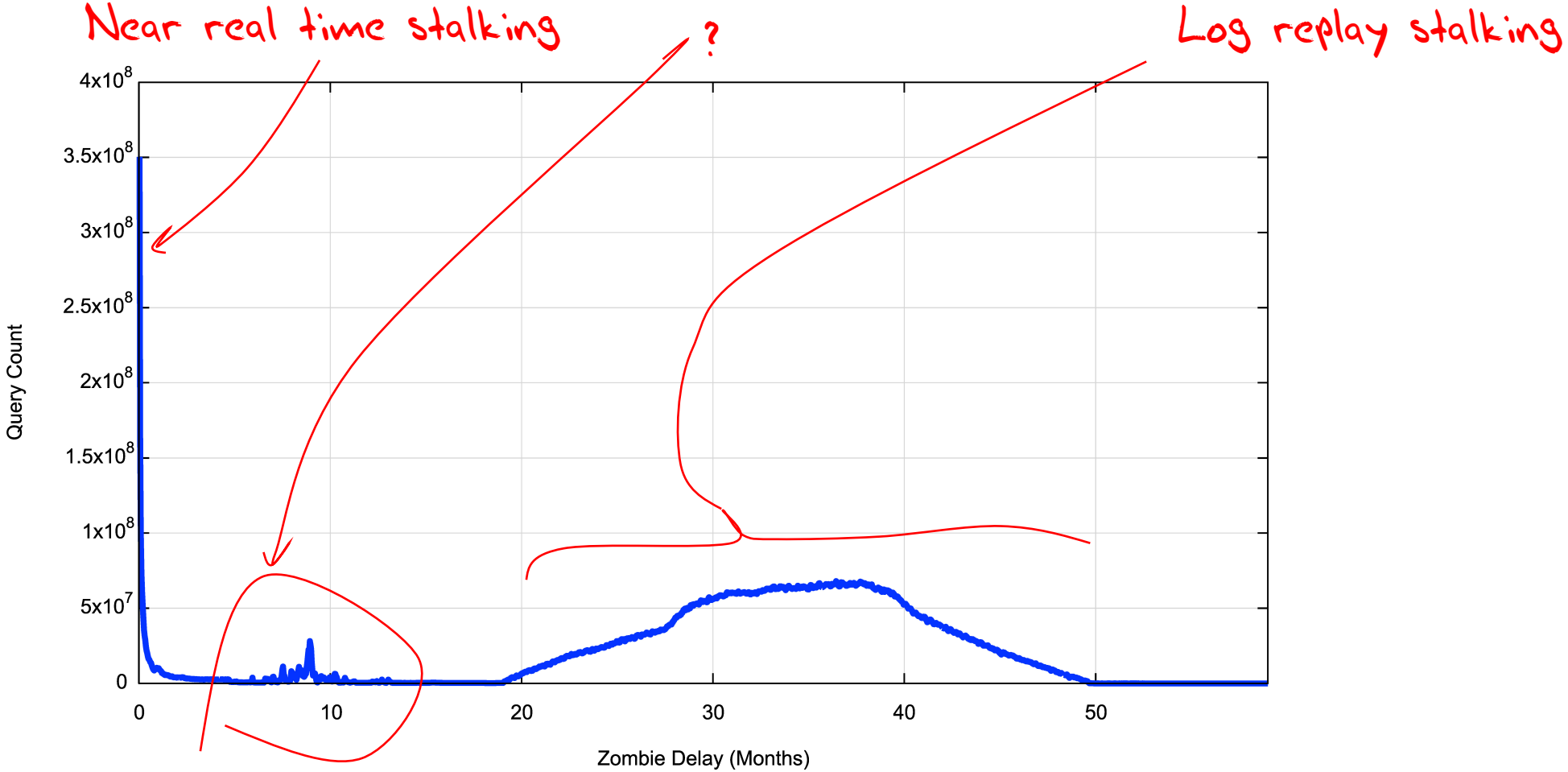
DNS Zombies

- There are two kinds of DNS zombie behaviours
  - Rapid tracking zombies that appear to track users in real time
  - Bulk replay zombies that replay DNS queries at a very high rate in bursts



DNS Stalking by Query Age

DNS stalking uses both really recent data and more than year-old data!



Top Stalker Origin Networks

	ASN	Query Count	AS Name
1	15169	1,513,141,306	GOOGLE, US
2	797	42,158,262	AMERITECH-AS, US
3	4837	37,319,869	CHINA169-BACKBONE CHINA UNICOM China169 Backbone, CN
4	28573	19,137,885	CLARO S.A., BR
5	38266	18,055,135	VODAFONE-IN Vodafone India Ltd., IN
6	45271	17,437,608	ICLNET-AS-AP Idea Cellular Limited, IN
7	55836	16,267,407	RELIANCEJIO-IN Reliance Jio Infocomm Limited, IN
8	4134	10,983,888	CHINANET-BACKBONE No.31,Jin-rong Street, CN
9	7922	10,196,512	COMCAST-7922, US
10	16509	9,872,862	AMAZON-02, US
11	14618	6,146,358	AMAZON-AES, US
12	2860	6,137,468	NOS_COMUNICACOES, PT
13	9808	5,797,160	CMNET-GD Guangdong Mobile Communication Co.Ltd., CN
14	36692	4,551,638	OPENDNS, US
15	327931	4,220,192	Optimum-Telecom-Algeria, DZ
16	13335	3,213,443	CLOUDFLARENET, US
17	3462	3,095,662	HINET Data Communication Business Group, TW
18	24445	2,804,270	CMNET-V4HENAN-AS-AP Henan Mobile Communications Co.,Ltd, CN
19	13238	2,653,073	YANDEX, RU
20	3303	2,254,898	SWISSCOM Swisscom (Switzerland) Ltd, CH
21	12322	2,166,796	PROXAD, FR
22	6128	2,141,747	CABLE-NET-1, US
23	7018	1,957,608	ATT-INTERNET4, US
24	25019	1,896,885	SAUDINETSTC-AS, SA
25	6799	1,823,699	OTENET-GR Athens - Greece, GR

Data gathered across April 2020

This data set is just a tiny glimpse into the overall pattern of DNS log capture and replay activity

DNS Surveillance

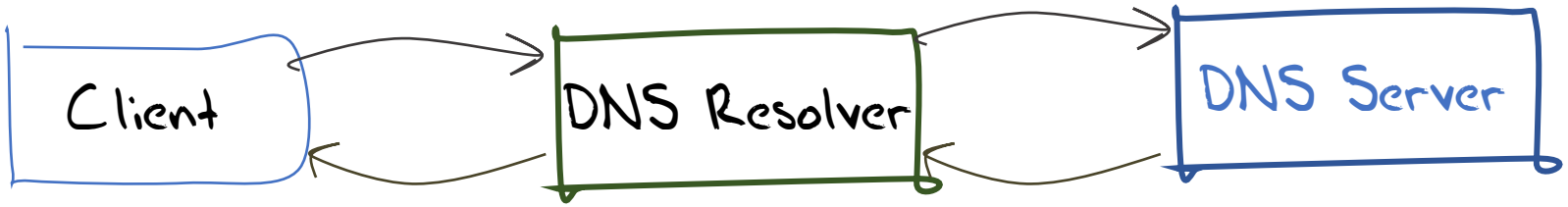
- Can we stop DNS surveillance completely?
  - Probably not!

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

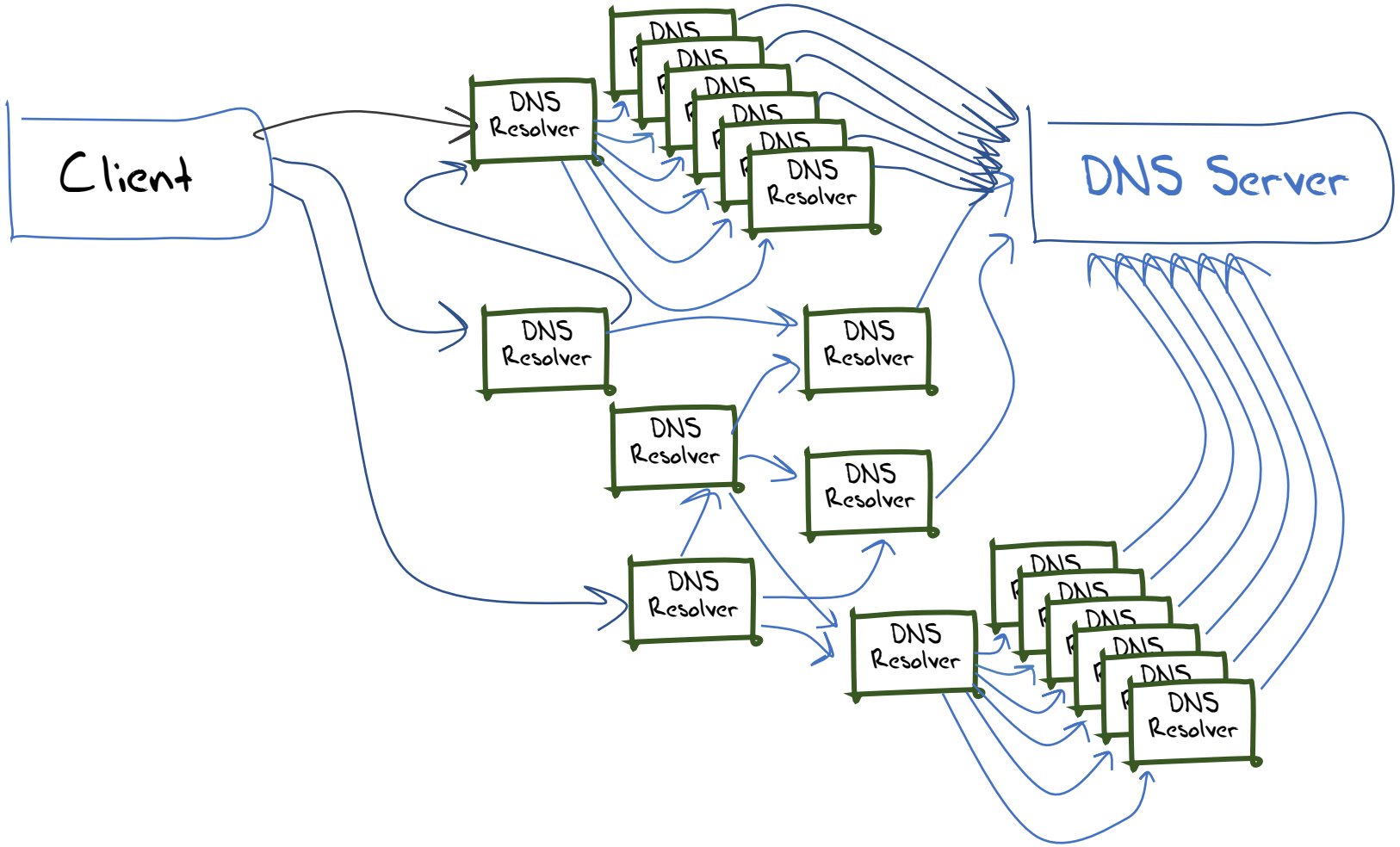
The DNS Privacy Issue

- Lots of actors get to see what I do in the DNS
  - My operating system platform
  - My ISP's recursive resolver
  - Their forwarding resolver, if they have one
  - Authoritative Name servers
  - Snoopers on the wire
- Can we make it harder for these "others" to snoop on me?

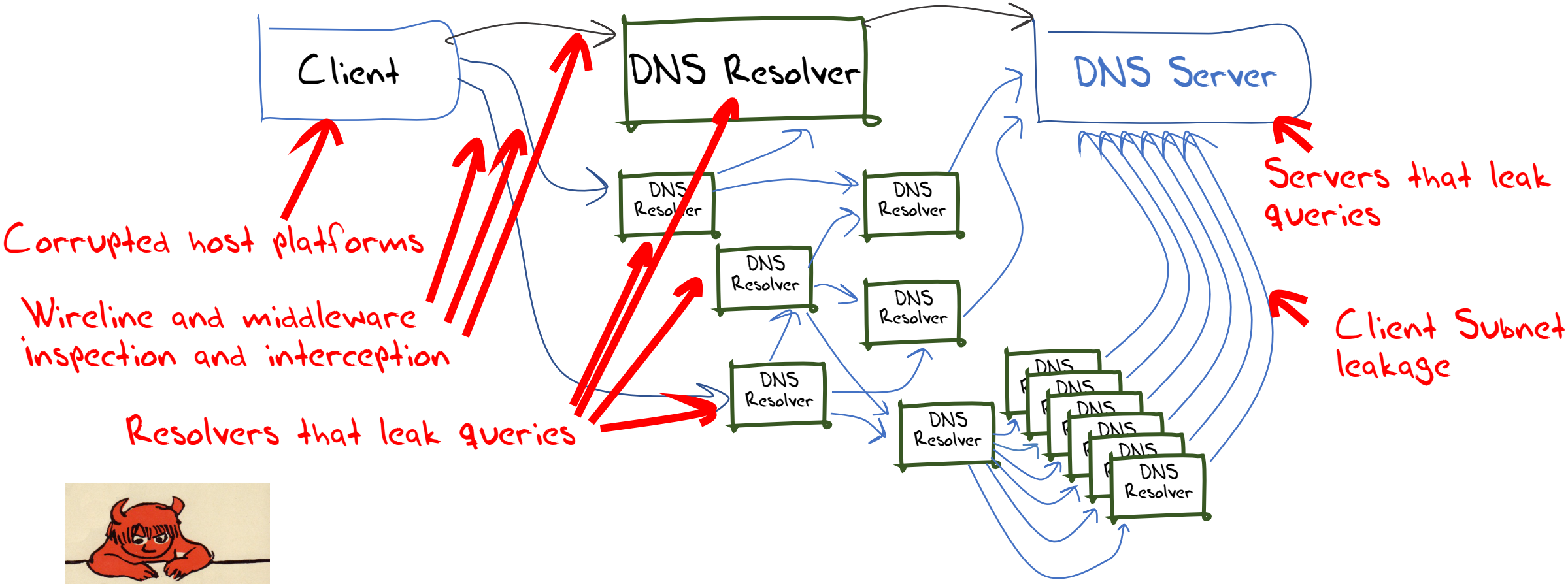
How we might think the DNS works



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 for users to trace**
  - Noone knows exactly where their queries go
  - Noone can know precisely where their answers come from
- The DNS is **used by everyone**

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

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 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

That's a delegated zone, so here are the servers for .com

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

That's a delegated zone, so here are 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

Example of QNAME Minimisation

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

Hi Root server, I want to resolve the A record for com
That's a valid name, so here are the servers for .com
Hi .com nameservers, I want to resolve the A record for example.com
That's a valid name, so here are the servers for example.com
Hi example.com server, I want to resolve the A record for www.example.com
The A record is 93.184.216.34

it's a good idea, but only some 3% * of users have their queries protected in this way today

* https://www.potaroo.net/ispcol/2019-08/qmin.html

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, etc
- These days interception systems typically offer an **incorrect response**

  (because no response invites re-queries and answers are cached longer than NXDOMAIN)
- How can you tell if the answer that the DNS gives you is the genuine answer or not?

DNSSEC

- DNSSEC is defined in RFCs 4033, 4034 & 4035
  - Adds a number of new RRtypes that allow a digital signature to be attached to RRsets in a zone and to define how keys that are used to generate signatures are also stored in the zone
- DNSSEC validation of the 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

  It will set the AD bit in the resolver response to indicate validation success

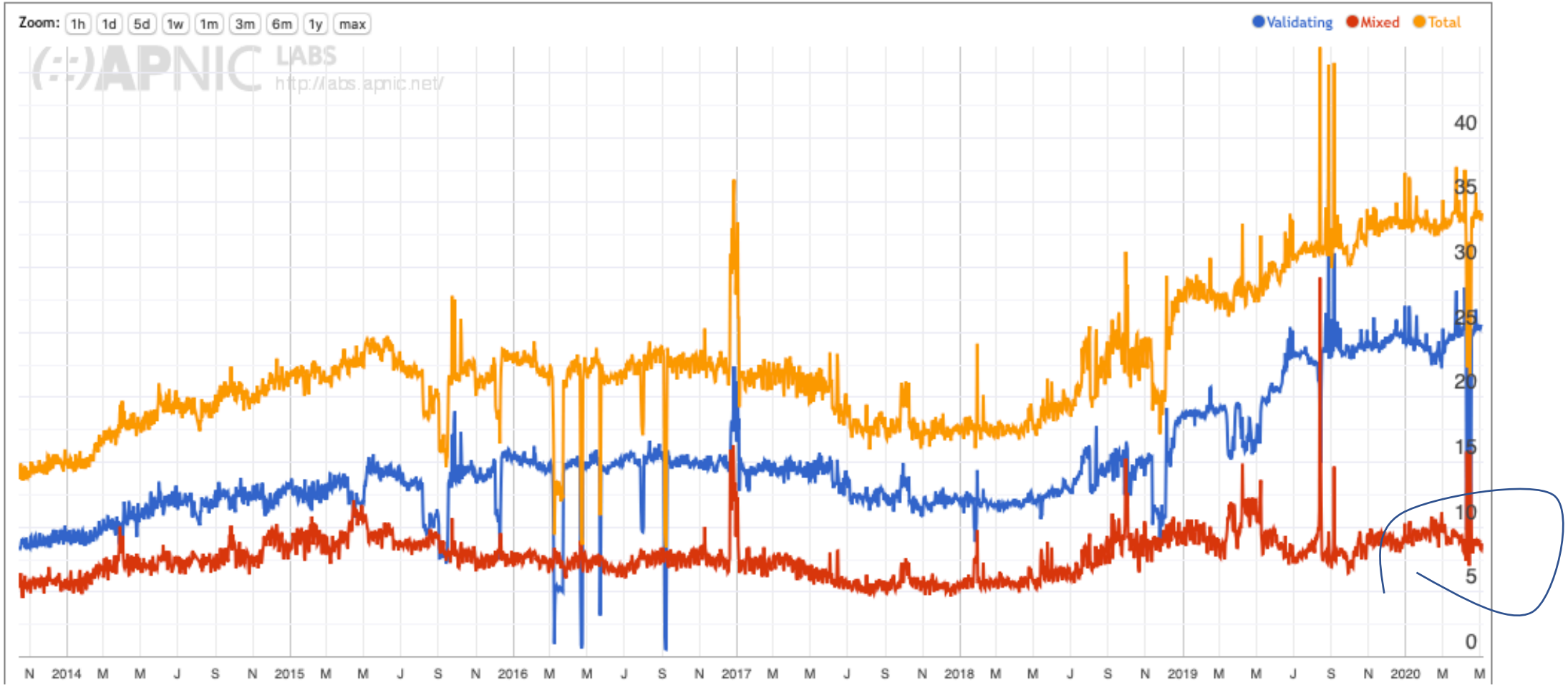
  Otherwise it will return SERVFAIL

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

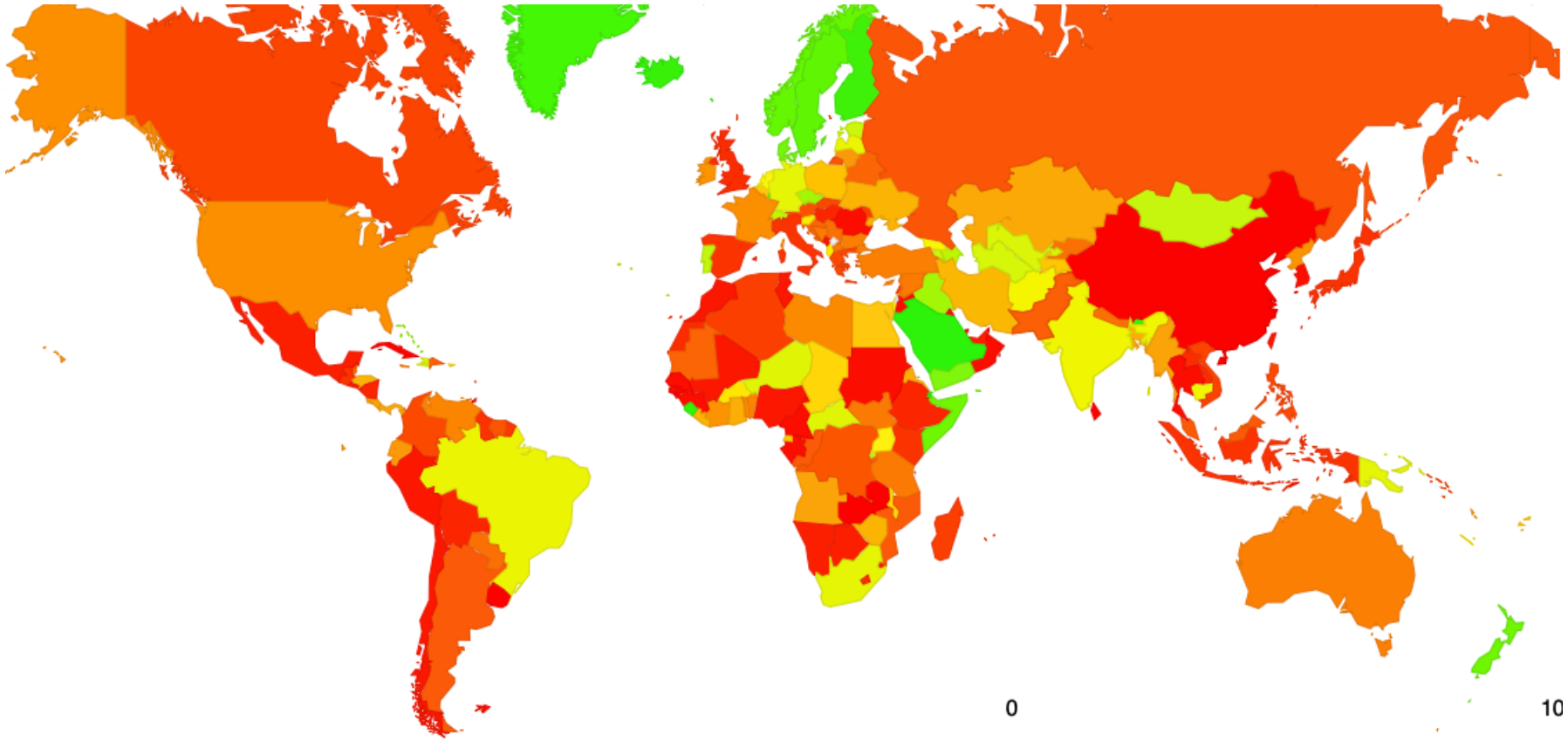
DNSSEC and Recursive Resolvers

- If you are going to use a DNSSEC-validating recursive resolver
  - Such as 1.1.1.1, 8.8.8.8, 9.9.9.9 or any other validating open resolver
- Then make sure that **all** your resolvers perform DNSSEC validation if you don’t want to be mislead
  - Because SERVFAIL from a validating resolver means “try the next resolver in your resolver list”

DNSSEC in Today's Internet



DNSSEC in Today's Internet

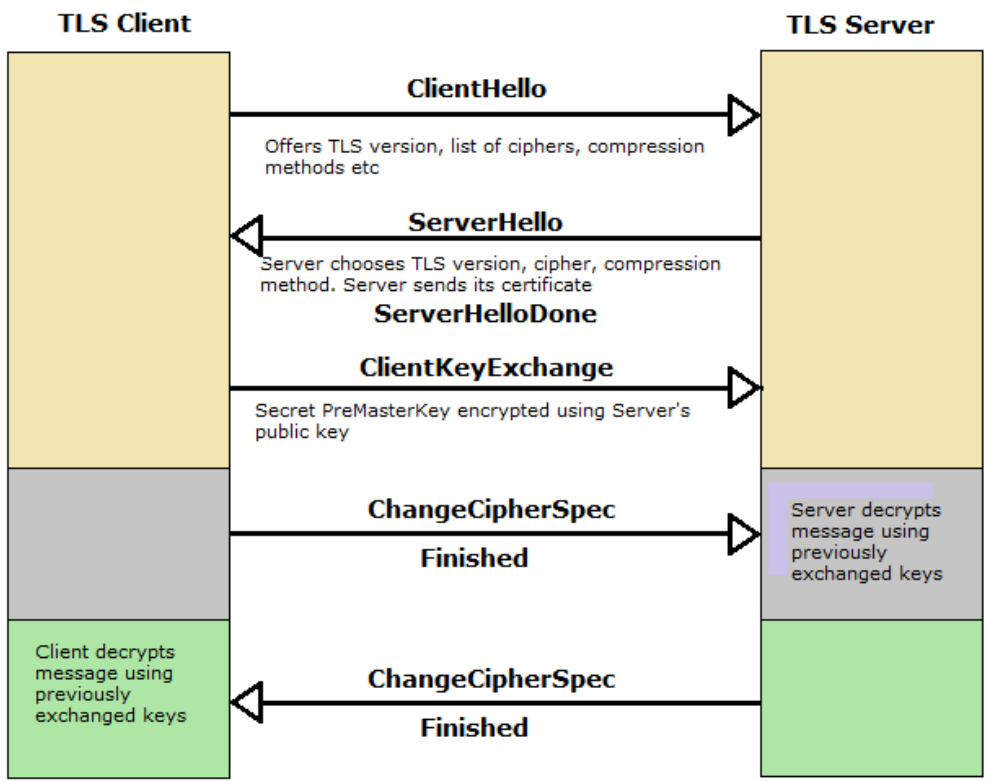


3. Middleware and WireTapping

- Protecting the content of DNS responses is part of what we need to make the DNS more robust
- If we want to prevent DNS inspection we also should look at encrypting 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
  - It's more challenging to use encryption between recursive resolvers and authoritative servers

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
- RFC 7858, RFC 8310



TLS 1.2 handshake

DoT

- Similar to DNS over TCP:
  - Open a TLS session with a recursive resolver
  - Pass the DNS query using DNS wireline format
  - Wait for the response
- Can use held DNS sessions to allow the TLS session to be used for multiple DNS queries
- The queries and the responses are hidden from intermediaries
- The client validates the recursive resolver’s identity

DoT and Android

Developers Blog DESIGN DEVELOP DISTRIBUTE

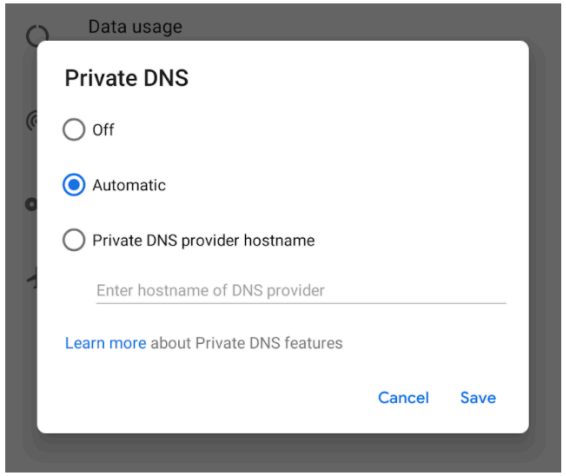
developer.android.com

Android Developers Blog

The latest Android and Google Play news for app and game developers.

DNS over TLS in P

The Android P Developer Preview includes built-in support for DNS over TLS. We added a **Private DNS** mode to the Network & internet settings.



By default, devices automatically upgrade to DNS over TLS if a network's DNS server supports it. But users who don't want to use DNS over TLS can turn it off.

Users can enter a hostname if they want to use a private DNS provider. Android then sends all DNS queries over a secure channel to this server or marks the network as "No internet access" if it can't reach the server. (For testing purposes, see this community-maintained list of compatible servers.)

https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html

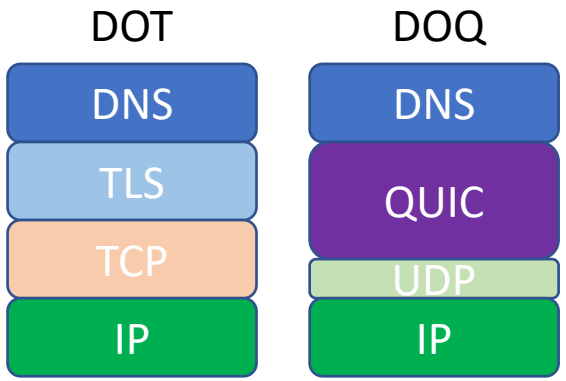
DoT

- Will generate a higher recursive resolver memory load as each client may have a held state with one or more recursive resolvers
- 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
- Supported by Bind (stunnel), Unbound, DNSDist

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
- The essential difference between DOT and DOQ is the deliberate hiding of the transport protocol from network middleware with the use of QUIC
- No known production implementations of DNS over QUIC exist, though IETF work continues

  draft-huitema-quic-dns-quic



DNS over HTTPS (DoH)

- DNS over HTTPS
- 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 wire format or JSON format DNS payload

DoH - 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

DoH - DNS within the Browser

mozilla wiki

Page | Discussion | Read | View source | View history | Search

Main page
Product releases
New pages
Recent changes
Recent uploads
Random page
Help

How to Contribute
All-hands meeting
Other meetings
Contribute to Mozilla
Mozilla Reps
Student Ambassadors

MozillaWiki
About
Team
Policies

Trusted Recursive Resolver

Firefox provides an optional resolver mechanism using a dedicated DNS-over-HTTPS server.

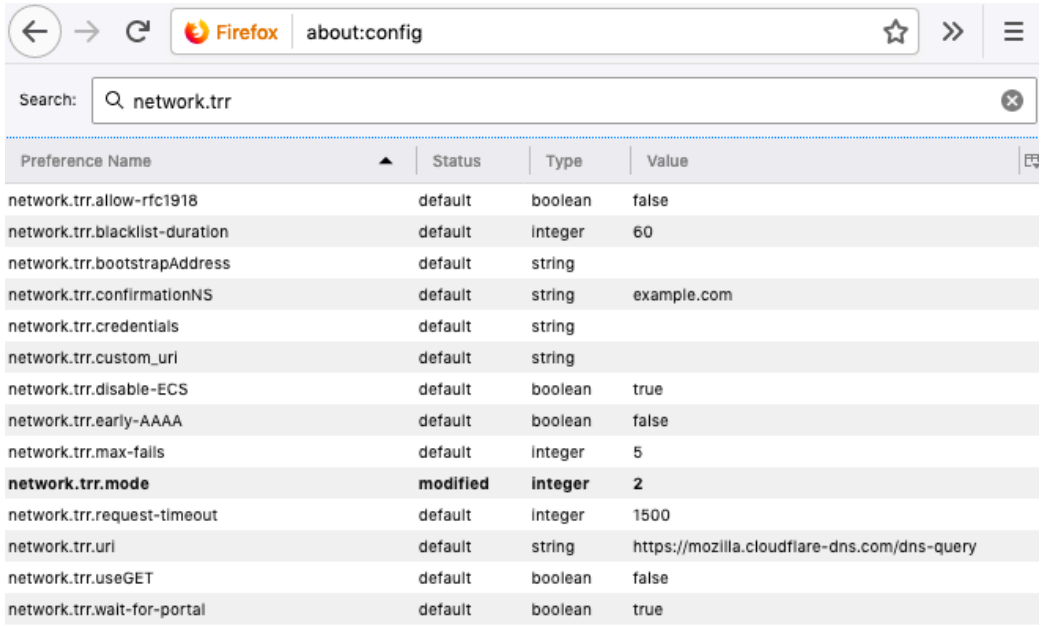
DNS-over-HTTPS (DOH) allows DNS resolves with enhanced privacy, secure transfers and improved performance.

Setting DNS-over-HTTPS in Firefox

- Set `network.trr.mode` to 2 to make DNS Over HTTPS the browser's first choice but use regular DNS as a fallback (0 is "off by default", 1 lets Firefox pick whichever is faster, 3 for TRR only mode, 5 to explicitly turn it off).
- Set `network.trr.uri`. Ones that you may use: https://mozilla.cloudflare-dns.com/dns-query (Privacy Policy), https://dns.google.com/experimental

TRR is preffed OFF by default and you need to set a URI for an available DOH server to be able to use it. Since the URI for DOH is set with a name itself, it may have to use the native resolver for bootstrapping. (Optionally, the user can set the IP address of the DOH server in a pref to avoid the required initial native resolve.)

All prefs for TRR are under the "network.trr" hierarchy.



Preference Name	Status	Type	Value
network.trr.allow-rfc1918	default	boolean	false
network.trr.blacklist-duration	default	integer	60
network.trr.bootstrapAddress	default	string	
network.trr.confirmationNS	default	string	example.com
network.trr.credentials	default	string	
network.trr.custom_uri	default	string	
network.trr.disable-ECS	default	boolean	true
network.trr.early-AAAA	default	boolean	false
network.trr.max-fails	default	integer	5
network.trr.mode	**modified**	**integer**	**2**
network.trr.request-timeout	default	integer	1500
network.trr.uri	default	string	https://mozilla.cloudflare-dns.com/dns-query
network.trr.useGET	default	boolean	false
network.trr.wait-for-portal	default	boolean	true

5. EDNS(0) Client Subnet

- There is a current debate between CDN operators that rely on customized DNS responses to perform content steering, and the use of large scale open resolvers that so not necessarily preserve locality
- The result is that the CDN operators wanted the client's subnet embedded in the query to ensure that the CDN could provide enhanced content steering for the client
- 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

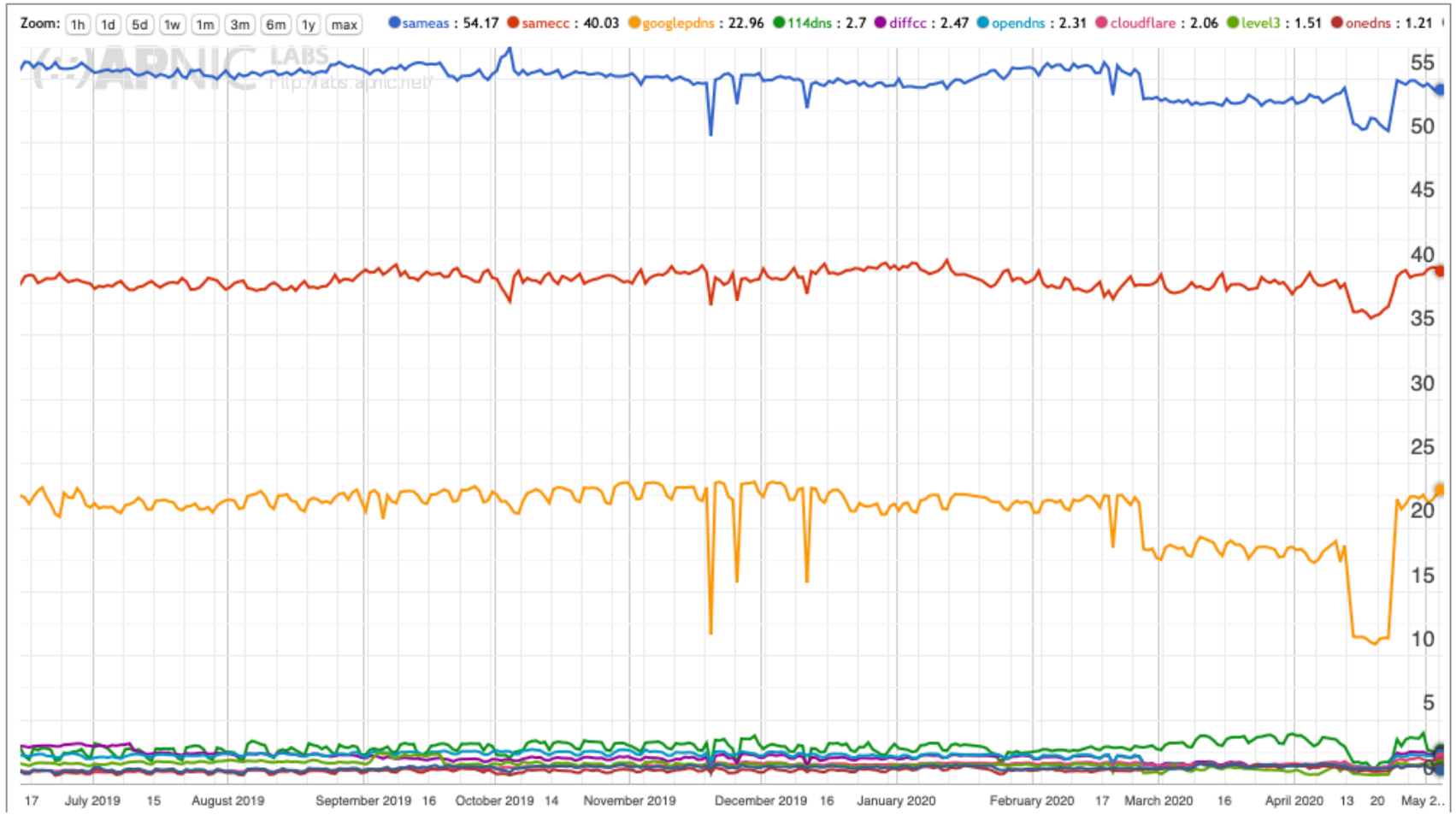
6. Hiding in the Crowd

- 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 Goolge, Open DNS, Cloudflare, Quad9 with your DNS then its far harder for anyone else to see you
    - But that is a very large amount of trust you are investing here!

Hiding in Google's crowd

Internet Measurements

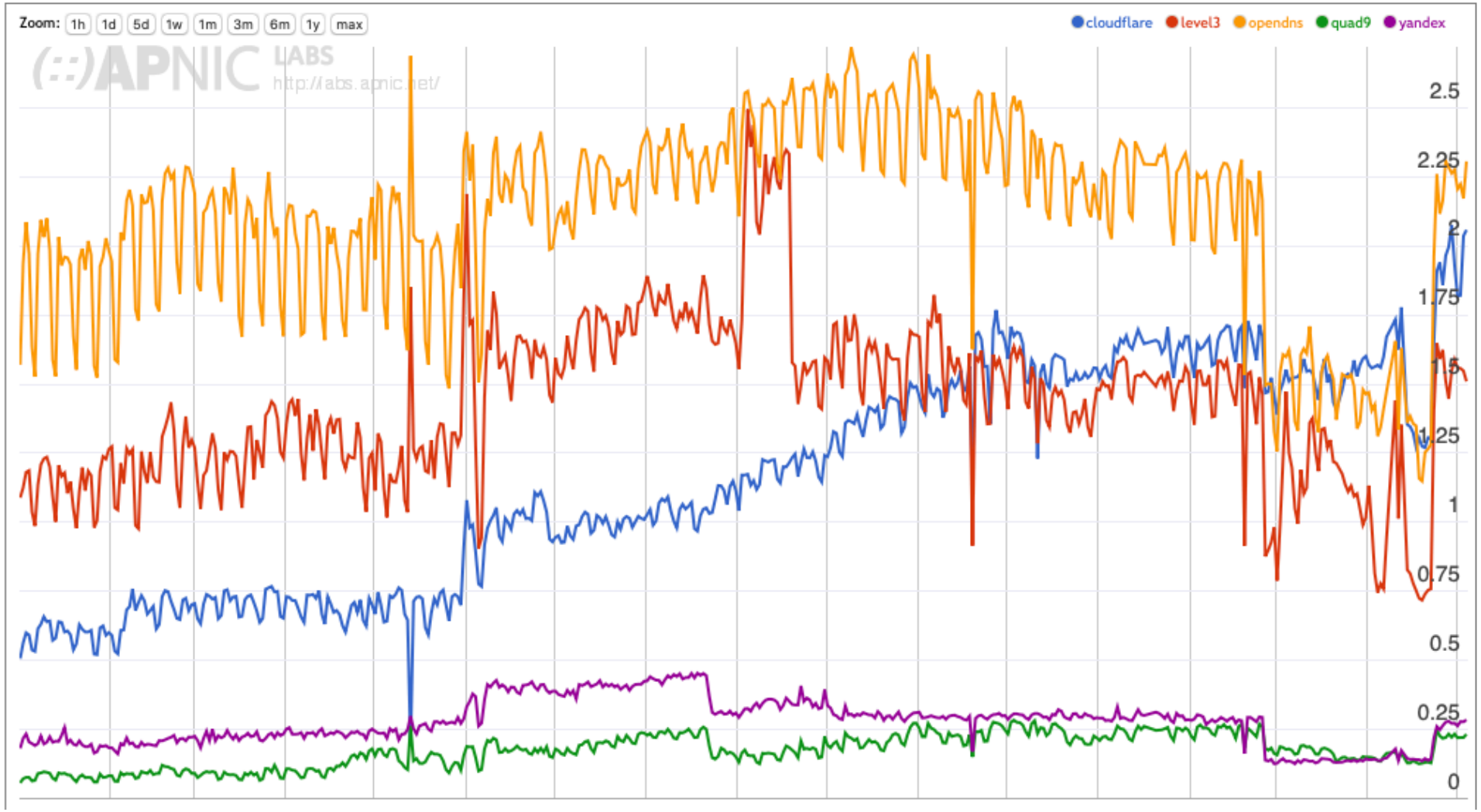


Only 52% of the world's users use their local ISP's DNS service

40% of users use in-country DNS resolvers

22% of the world's users use Google's public DNS

Hiding in crowds that are not Google's



Can't see me!

- Run you own recursive resolver
  - Can DNSSEC validate directly
  - All queries in the open
  - Authoritative servers see me
  - Can do Qmin directly
  - Can disable ECS
  - Limited caching = slower!
- Use ISP's resolver
  - All queries in the open
  - ISP resolver can track me
  - I trust the ISP resolver to DNSSEC validate
  - Authoritatives can't see me
  - No control over Qmin
  - No control over ECS

Still can't see me!

- Use an Open Resolver with DOH
  - DNS transactions not visible locally or to ISP
  - I trust the open resolver to DNSSEC validate
  - Authoritative servers can’t see me
  - I expose my behaviour to one open resolver but no others
  - No control over ECS
  - Blockable by destination address filtering
- Random selection across a number of Open Resolvers with DoH, with local validation
  - DNS transactions not visible locally or to ISP
  - DNSSEC validation locally
  - Authoritative servers can’t see me
  - No single open resolver sees my entire activity profile
  - No control over ECS
  - Blockable by destination address filtering

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 that recursive resolver operator
- Local DNSSEC validation is always a good idea, even though there is a time penalty

But do you have a choice any more?

My (current) DNS Config

I use DNS over HTTPS

- I have configured Cloudflare's Cloudflared * to listen of localhost:53
- I have set up my local /etc/resolv.conf to contain 127.0.0.1
- All my DNS queries leave my laptop in an HTTPS port 443 stream towards 1.1.1.1

* https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy/

Thanks!