June 2020
Geoff Huston

Where is the DNS Headed?

I was on a panel at the recent Registration Operations Workshop (http://regiops.net) on the topic of DNS Privacy and Encryption.

The question I found myself asking was: "What has DNS privacy to do with registration operations?" The registration function is part of the process of public attestation relating to some form of title of exclusive control. But the name registration entry has very little if anything to do with the manner of resolution of that name. It should not matter in the least if a client uses plain DNS, DNS over TLS, DNS over HTTPS or DNS over QUIC to pass a query to a recursive resolver. Equally it should not matter in the least to the registry operator of the recursive resolver uses an encrypted tunnel to query an authoritative server. Similarly, it shouldn't matter to the registry operator if the client uses Query Name Minimisation or not. Of all the things a registry operator may or should concern themselves about, there is little to find in DNS privacy and encryption that touches upon their operation. Why is this topic even on the agenda of a registration operations workshop?

But perhaps this is a little too simplistic. Perhaps there are some elements of the way in which DNS resolution is being used in today's network that point to further changes that may impinge on the name registration function and the coherence of the Internet's name space as a whole. Let's see if we can build a credible case that privacy and encryption have the potential to facilitate fragmentation on the DNS namespace.

To start this, let's take a step back and look at "traditional" DNS name resolution (Figure 1).

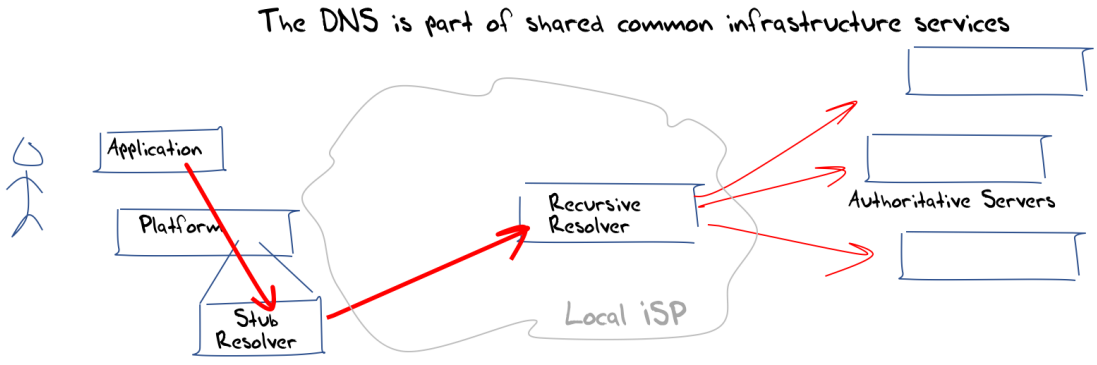

Figure 1 - Traditional DNS Resolution

In this model the DNS is part of the common infrastructure of the Internet. Applications are not necessarily aware of the nature of name resolution. The application calls the host platform operating system through a library call such as `gethostbyname` and at that point the process of name resolution becomes a common process used by all applications on this platform. If the name is not in a local cache of recently resolved names, and if the name is not locally defined in `hosts.txt` (yes, many platforms still support this now archaic form of name resolution) then it will pass the query on to a

recursive resolver to resolve the name. Which resolver? In this model of classic DNS the resolution function is performed as part of the local network service. When translated into the context of ISPs, the name resolution function is performed by the local ISP.

The application has no role in the name resolution process, other than triggering the initial query to resolve a name, and the name resolution process is completely independent of the application. Any application on the same platform could ask the same query and the DNS response would presumably be the same. Names are part of the common network infrastructure in this model.

It is interesting to ask how prevalent this model is in the Internet today. Do end clients use their locally provisioned DNS recursive resolver? We can answer this question, or a slight variant of this question using measurement data collected by APNIC Labs. What proportion of clients use a resolver in the same network as the client itself to query authoritative servers? And is this rising or falling over time? Figure 2 shows the data that can answer these questions.

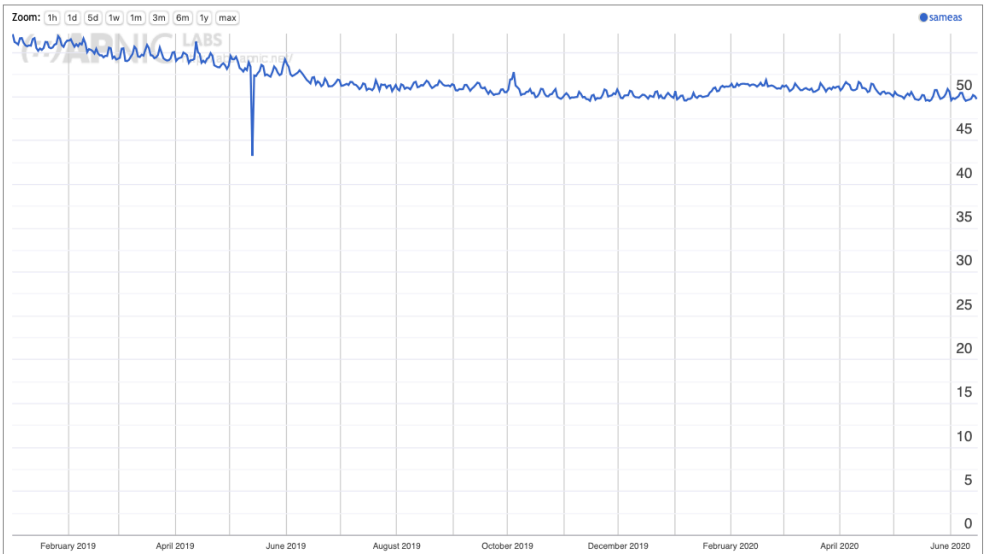

Figure 2 – Use of Same AS Resolvers

It appears that slight under one half of users use resolvers operated within their local network where the resolver directly processes the request without forwarding it to other external resolvers. Over the past 18 months this number has fallen by some 5%. The local of local resolvers is slightly more prominent on weekends while weekday use tends to use non-local resolvers.

The conclusion is that this local infrastructure model of the DNS is undergoing some changes. What is changing? One major factor here is the introduction of the use of open recursive resolvers. This class of resolvers include services operated by Google's 8.8.8.8 service, Open DNS, Cloudflare's 1.1.1.1 service, Quad 9 (9.9.9.9) and others. In this model the DNS recursive resolve r is provided as a distinct service and the DNS name resolution function is lifted up to be an overlay service that operates in a way that is analogous to any other Internet application service (Figure 3).

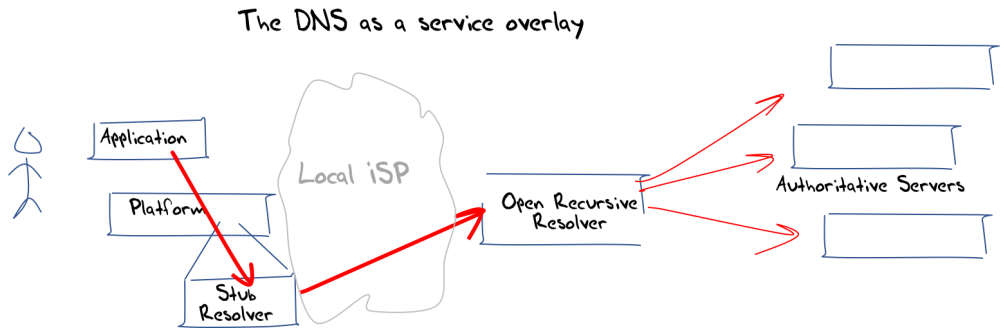

Figure 3 – DNS as an overlay application

How prevalent is this overlay service model in today's Internet? What proportion of users pass their DNS queries to open DNS resolvers?

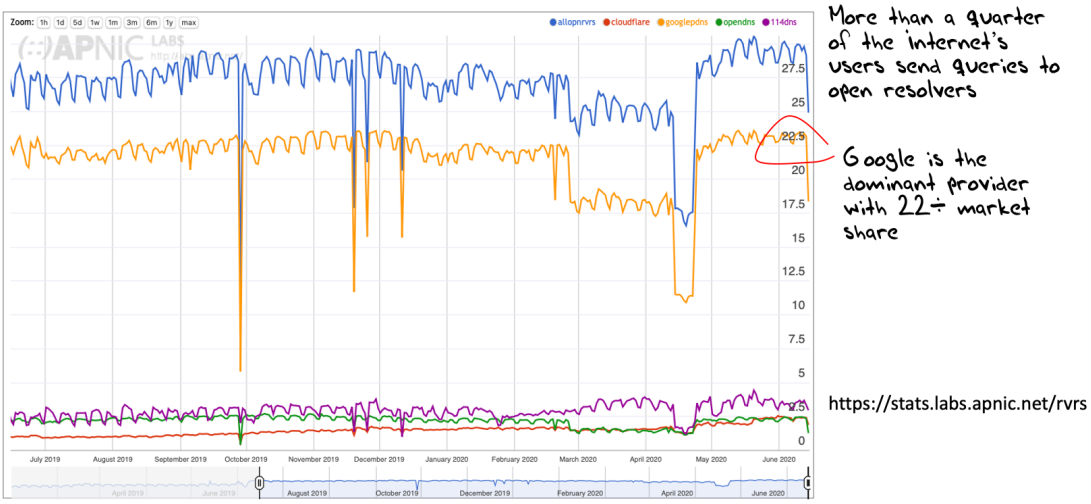

Figure 4 – Use of Open Resolvers

The answers to these questions are shown in Figure 4. Some 28% of users send DNS queries via open resolvers. This is more prominent on weekdays, suggesting that enterprise networks are more inclined to use open resolvers than consumer service networks. The open resolver market share is not uniformly distributed, as some 22% of users use Google's public DNS service. It is unlikely that this is the result of individual configuration of sub resolvers. More likely is the use of forwarder functions in network-based resolvers, that pass all client queries onto Google's service. This may be a choice driven by cost, service reliability, data integrity or many other potential reasons, but the outcome is the same. The DNS is increasingly looking like an overlay service and not common infrastructure.

Let's now bring privacy and encryption into this picture. The DNS is certainly a rich field of activity. DNS queries are in the open, and can be readily intercepted and fake responses substituted. This DNS interception is not just part of the toolset for malware, but an intrinsic component of many national internet filtering measures. It can be observed that many national content policies are implemented through DNS interception. It's not just removing certain names from the purview of users' activities. The DNS is a rich source of data about users and applications. If an observer were to collate the complete set of DNS queries from an individual user over a period of time then it's possible to construct a very accurate profile of that individual user. It seems that the DNS is a willing collaborator in this exercise of digital surveillance and control (Figure 5).

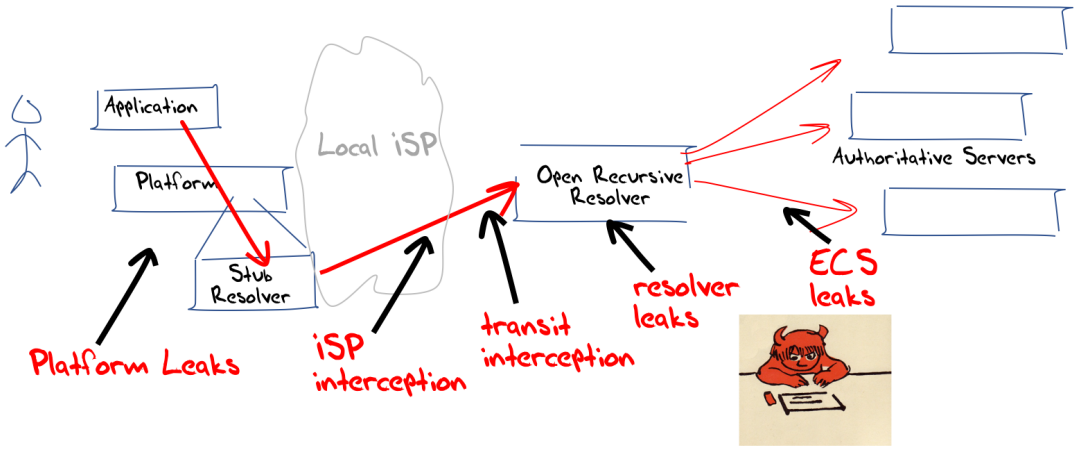

Figure 5 – DNS Interception and surveillance

At almost every step of the DNS resolution process there is the opportunity to link a DNS query to the original end user, and the opportunity to intercept a query and substitute a synthetic answer.

These days we talk of "informed consent". It's ok for a user to grant permission to have their data collected in various ways, but the user's explicit consent is usually a necessary precondition, and such consent needs to be informed as to the purposes to which such personal data will be used. All well and good, but how can a user prevent the unconsented collection of such data. How can a user stop third parties from tapping into the DNS and gathering, and even altering, DNS data without the users' explicit consent?

It's here that DNS privacy and encryption enter into the picture. Privacy measures include restricting the level of "chattiness:" of the DNS, and stopping exposing the full query name to all authoritative name servers through Query Name Minimisation. But the most effective tool is encryption. Major steps have been taken with the web in recent years. We've seen the introduction of free domain name certificates that have turned session level secure services from an expensive and exotic luxury good to a universally accessible commodity. The introduction of Transport Layer Security (TLS) has transformed the web. Now almost every HTTP session is in fact an HTTPS session using TLS. (Figure 6)

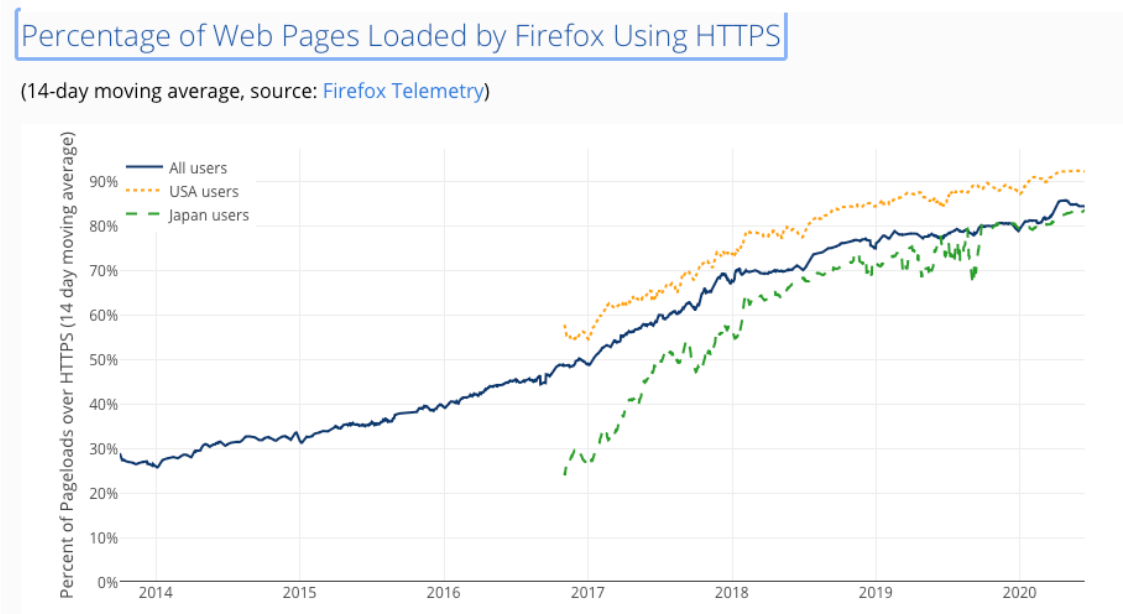

Figure 6 - Usage of HTTPS – from https://letsencrypt.org/stats/#percent-pageloads

If TLS has worked so well for the web, then why not do the same for the DNS? Encryption of the query on its path from the user's device to the recursive resolver would prevent both surveillance and interception of the DNS. The fundamental change is a shift from a datagram UDP protocol to a session protocol, but the relationship between a stub resolver and its recursive resolver can amortise the overheads of setting up a TLS session over TCP across entire sequences of stub queries, so for the stub resolver this is a measure with few direct disadvantages. The recursive resolver is placed under a greater load to maintain session state and the shift from UDP to DNS over TLS, or DoT, has some cost implications for the resolver. However, if we can make it work for web transactions, and we have, then there seems no reason why this shouldn't work for the DNS recursive resolver.

As shown in Figure 7, one approach to improve the security of the DNS is to encrypt the path between the local onboard stub resolver and the remote recursive resolver, using TLS as a wrapper for DNS queries. Two very widely used mechanisms for interception and surveillance, trapping queries as they pass across the network between the stub and the recursive resolver are prevented through the use of TLS.

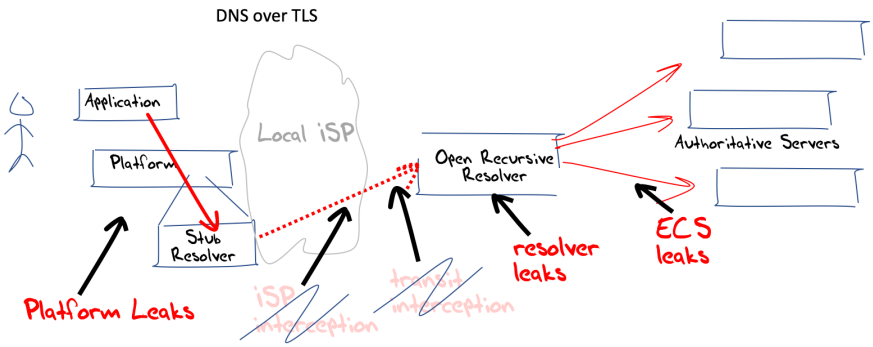

Figure 7 – DoT: DNS over TLS

Why stop at TLS? What if we added a layer of HTML as well? In one sense it's a minor change and the behaviour should be similar. But in another sense, it's a big jump. If the DNS queries and responses are wrapped in HTTPS then any HTTP engine, namely a browser, can undertake DNS queries and get responses all on its own. Why would a browser want to do this?

Firstly, it can prevent the platform from peeking into the application's activities. The DNS transactions are secured from eavesdropping from the application itself.

Secondly, its possible to use HTML to add additional behaviours to name resolution. HTML supports push functions where objects are pushed to the client by the server. It's possible to treat DNS responses in a similar way and reduce user wait times by pre-provisioning the user with name resolution outcomes that a server may know that the client will likely ask.

Thirdly its possible to customeise the answer by adding HTML attroibutes to the query. The path through EDNS(0) has been long and tortuous and the outcomes so far, namely Client Subnet are not exactly the best of outcomes. Quite the opposite in fact. So if DNS queries can be made in HTML then why not communicate metadata about the query in HTML as well?

Now it's not only HTTPS that can be treated in this way. QUIC can be used in an analogous manner, and this leads to a generic approach to sealing up the client-side DNS from interceptors and eavesdroppers. (Figure 8).

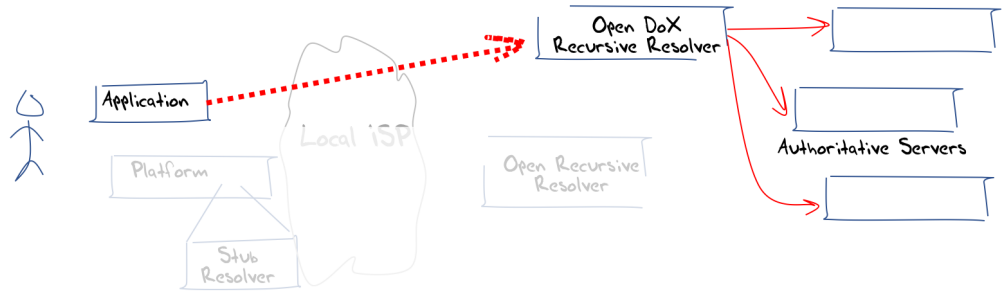

Figure 8 – Application-based DNS over HTTPS/QUIC

If each application is free to communicate with its own DNS name resolver, then it's a very short step to use an application-specific name resolution service. In this way the application can embed resolution of its own application-specific handles and identifiers. Its not particularly necessary for the application to frame queries in the DNS to pass queries to an application-specific server, nor is it strictly necessary to limit the queries to be delegated DNS names in what we used to call the "global DNS. Each application is able to customise a namespace to meet its own requirements. Thre result is Figure 9, which is a scenario that is well removed from the original DNS ad infrastructure starting point.

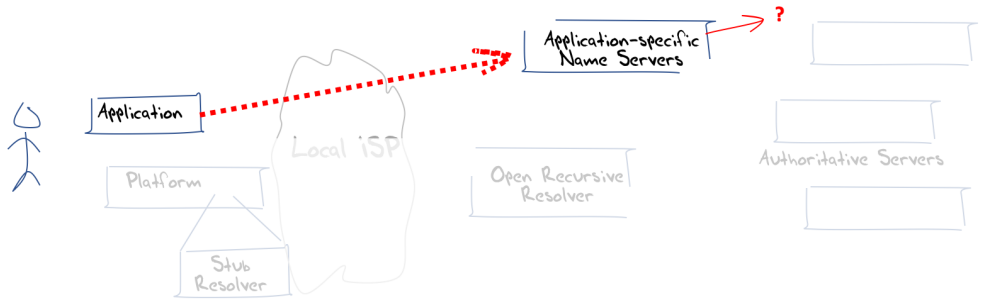

Figure 9 – Application-based name services

At this point we have left the concept of DNS as a common infrastructure far behind. These mechanisms to secure the transactions and remove them from open view are accompanied by a shift to move the name resolution function to one that is under the application's direct control. Once this shift to application specific services is made the need to use a common DNS name space recedes. Applications can then determine whether they want to use various forms of customisation to improve the performance of the application and augment the namespace with their own bright ideas.

What does this new application-based set of name spaces really look like? Is it still cohesive, or has it been comprehensively fragmented?

If the characterisation of what makes the internet a single network is a single address space and a single name space, then it's pretty clear that we've dispensed with a coherent and uniform address plan already as NATs are just so pervasive. But if a coherent name space is all that's left of a single unifying Internet what happens when we tear that apart as well?

Back to that original question: "What has DNS privacy to do with registration operations?" If the registry function is a function that preserves the coherence and integrity of a single embracing name space for the Internet, then our efforts in DNS privacy are leading us in directions that threaten to rip all of this apart!

If this application-level fragmentation of the namespace is where the DNS is headed, then it's unclear to me what comes next!

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net