The ISP Column
A column on various things Internet

October 2018
Geoff Huston

DOH!

If you had the opportunity to re-imagine the DNS, what might it look like?

Normally this would be an idle topic of speculation over a beer or two, but maybe there's a little more to the question these days. We are walking into an entirely new world of the DNS when we start to think about exactly might be possible when we look at DNS over HTTPS, or DOH.

We are now in a position to look at the standardization work in the IETF where the DOH Working Group is attempting to standardize DNS over HTTPs (https://tools.ietf.org/html/draft-ietf-doh-dns-over-https). The effort is directed to standardizing encodings for DNS queries and responses that are suitable for use in HTTPS, enabling a standard and interoperable mechanism for DNS names to be resolved over secure TCP connections using the HTTP/2 protocol. It's a hybrid approach that attempts to integrate standard HTTP methods, error codes, and other semantics to the greatest extent possible, while still preserving the query response nature of the 'traditional' DNS name resolution protocol and the DNS resolution protocol format.

The primary focus of this working group is to develop a mechanism that provides confidentiality and connectivity between DNS clients (e.g., operating system stub resolvers) and recursive resolvers. In this way DOH is a direct substitution for the DNS over UDP step that allows a client-side stub resolver to pass queries to a recursive resolver and receive responses, where the IP and UDP wrappers are replaced with HTTPS wrappers.

What approach are they using here?

New URI labels for different URI name schemes are unfashionable these days, and the DOH work is no exception. Rather than recycle the earlier `dns:` URI prefix, the DOH approach uses the currently preferred mechanism, namely the use of a well known URI path of "`/dns-query`". In this scheme a DNS query would look like:

```
   :method = POST
   :scheme = https
   :authority = dnsserver.example.net
   :path = /dns-query
   accept = application/dns-message
   content-type = application/dns-message
   content-length = 33

   <33 bytes represented by the following hex encoding>
   abcd 0100 0001 0000 0000 0000 0377 7777
   0765 7861 6d70 6c65 0363 6f6d 0000 0100
   01
```

This approach uses the existing binary DNS on-the-wire format, using a MIME data type of `application/dns-message`. When using the GET method the DNS query is encoded in Base64url, while the POST method places the binary DNS query in the body of the POSTed HTTP object, as per the above example.

While the IETF work in progress to standardize this technique uses the DNS on-the-wire format for the payload of DNS queries and responses, there are also implementations of DNS over HTTPS where the payload is formatted using JSON encoding, defined in RFC 8427 and denoted as `application/dns+json`. In all other respects these two approaches are identical, and the transformation between the one-the-wire format and a JSON format is very straightforward.

A DOH query and response using JSON formatting looks like:

```
$ curl -s -H 'accept: application/dns+json' \
    'https://dns.google.com/resolve?name=www.potaroo.net&type=A' | jq

{
  "Status": 0,
  "TC": false,
  "RD": true,
  "RA": true,
  "AD": true,
  "CD": false,
  "Question": [
    {
      "name": "www.potaroo.net.",
      "type": 1
    }
  ],
  "Answer": [
    {
      "name": "www.potaroo.net.",
      "type": 1,
      "TTL": 6399,
      "data": "203.133.248.2"
    }
  ],
  "Comment": "Response from 203.133.248.2."
}
```

The HTTP 2xx response code is used for all DOH responses, irrespective of the DNS response code in the payload. The DNS responses also use DNS on-the-wire binary format, carried as the payload of the HTTP object.

My first reaction to this work was to struggle to understand where the win is with DOH. The additional HTTP wrapping seems to add little other than extraneous window dressing, and apart from the cheap thrills in trying to transform the external appearance of one protocol into the guise of another I couldn't appreciate any advantages for this approach. If what we are after is the simple ability to conceal DNS queries within an encrypted channel, then the TLS part of DOH is doing all the heavy lifting here while the HTTP/2 component appears to be little more than a source of extraneous adornment. So why not just use DNS over TLS and move on?

DOT vs DOH

If the objective is to secure the communications between the stub resolver residing on the end user's system and their chosen recursive resolver then, in theory, one could take DNS over UDP with a fallback to TCP and replace it with DNS over DTLS with a fallback to TLS and preserve all other aspects of the protocol operation. However, DNS over DTLS appears to have little to recommend itself in this context, and if we want to support larger DNS payloads, including various forms of omnibus responses, then perhaps we should just head straight to DNS over TLS (DOT). In this approach we can factor in the overheads of TLS session establishment as a one-off cost that is spread across multiple queries and gain the benefit of reliable and secure communications with the recursive server. The server incurs some level of additional overhead in moving from a stateless responder architecture to a session-based system, but we seem to have managed this in the world of HTTPS, and perhaps the incremental cost of session state in the server is not that forbidding.

> One view is that most of the DNS server side dimensioning is not based on normal query load in any case, but based on the anticipation of DDOS attack. The capacity of a server includes the consideration of having sufficient capacity to absorb the hostile query traffic and continue to provide responses to non-hostile queries at the same time. With a change from the DNS transport from UDP to TCP the attack profile changes, and the capability to undertake UDP source spoofing goes away. Attacks will no doubt still take place, but the attack profile is expected to be reduced significantly with the change to TCP. This implies that the reckoning of the incremental server load to service TLS clients should include some factor of a reduced potential DDOS attack load profile.

But why add HTTP headers to the TLS session? What do we gain in terms of the capability and efficiency of name resolution transactions by taking this further step from DOT to DOH?

Privacy Resilience

DNS over TLS is certainly far more resilient than DNS over UDP to the various attempts to intercept, block and otherwise interfere with the transactions between a stub DNS resolver within an end system and its chosen recursive resolvers. Once a TLS session is established then the client can be assured that third parties cannot interfere with the flow of DNS queries and responses between the client and the recursive resolver. The client can also be assured (to the extent that such assurances are reasonable in the Domain Name PKI world of Certificate Authorities) that the client is actually communicating with the named recursive resolver.

But the use of TCP port 853 as a dedicated port does admit the possibility of third party blocking of DNS over TLS sessions. The use of unencrypted Server Name Indication field (SNI) when setting up the TLS session an observer could determine which recursive resolver is being used by the client.

DOH uses TCP port 443, allowing both Web and DNS transactions to take place within a single HTTP/2 session if the server is set up to handle both forms of service. Again, the SNI field, if unencrypted, does reveal to third party eavesdroppers the domain name of the DOH service used by the client. However, while a filter on TCP port 853 would block DNS over TLS without necessarily impacting on content services, the same may not be feasible with DOH. Techniques that combine DOH services with web services, as used by some Content Distribution Service providers already, would mean that attempts to block DOH traffic without impacting on potentially unrelated content would be far more challenging.

Pull vs Push

The current architecture of the DNS is one that is uncoupled from other application transactions. An application learns of a service name through some application or user interaction and then has to pass this name to a local DNS stub resolver for the translation of the name into a service address. The DNS itself is a "pull" response model, in that the DNS finds it difficult to anticipate the future name resolution requirements of an application. It's true that a diligent and highly attentive application could scan its incoming data stream, identify references to service names that will require DNS resolution, and commence the DNS queries immediately via its local DNS query sub-system. HTTP contains server push as well as user pull.

Can we utilize this by embedding the DOH URI into the web object as a push object? It would certainly be faster as the query is being provided before the client application makes an explicit DNS request. And if the resolved DOH object also included a DNSSEC chain response (RFC 7901) then the client could validate the proffered DNS data without having to make a DNS query at all!

Pandering to the Paranoid Application

Applications these days do not necessarily make extensive use of the library provided by the underlying host platform. An admittedly somewhat paranoid view of privacy would attempt to protect an application's activity not only from potential eavesdroppers in the network, but also from potential eavesdroppers sitting on the same host platform as the application. If a browser really wanted to keep your browsing activity a secret, then it makes sense to avoid using the platform's DNS services and instead channel the DNS requests back to a trusted resolver across an open HTTPS session.

DOH and Name Spaces

DOH raises an interesting question about visibility of names and the scope of the DNS name space.

There is a general assumption in the DNS that name space is uniformly available. If a URL works for you then you can pass this URL to me with the expectation that I will connect to the same content or service if I use this URL. The way we support this assumption is to set up name servers as promiscuous servers, where a server will respond to all queries in a similar manner, irrespective of who is making the query. This general observation is not quite the case when the DNS is used in content steering contexts, where the DNS response handed out by an authoritative server based on the server's assumption about the location of the client requesting name resolution. But notwithstanding the way in which answers are generated, the expectation is that names in the DNS are intended to be universally visible, and the name space is uniformly available.

However, a name may be defined within a web object that performs DOH-style push to clients does not need to have a counterpart in the query-based DNS as we understand it. If the name is signed with DNSSEC, there is no need to provide authenticated denial of existence, so there is no requirement to provide NSEC records for these server-pushed names. On the assumption that the only way to "see" that such a name exists is via this push mechanism within the HTTP space then a parallel conventional query-based DNS could provide a different definition of the same name, or even deny its existence. As long as the client application, such as a browser, uses DOH-pushed names solely in the context of the application then the client may be able to maintain the distinction between DNS names learned by explicit query to the DNS and application-context DNS names that were learned buy DOH-push. Why would anyone do this? With this approach each web page is capable of constructing a local name space that is essentially defined by that web page, and names can be 'bound' to that web context and may not be accessible via a general DNS query.

For some this may well look like a completely novel approach to naming, where names defined by DOH are tied into a referential context and do not necessarily exist outside that context. It challenges the characterization of the DNS as a single unified and uniform namespace as it allows the creation of private branching within the name space.

After more than twenty years of experience with the DNS we are probably justified in concluding that the remains a significant population of users that do not really like hierarchical namespaces. Not only have we seen many country code domains 'flatten' their namespace and allow direct registrations as second level names under the country code, but the same pressures to flatten the name space exist in the root zone itself. The ongoing rounds of release of generic TLDs in the root zone of the DNS space is proving that such releases of new names in the root zone make little inroad into the vast ocean of latent demand.

The enforced hierarchy was seen as a consequence of a single unified name space. Are we seeing in DOH the emergence of the possibility of a rather strange multiverse of name spaces, each defined by the encompassing web object and its embedded push DOH objects? If we were to go further and allow the web object to define itself as the root of trust in the names delivered by the web object and allow us to push DNSSEC chains that point to this trust point, then indeed the mythical nirvana of a flattened, yet extensively partitioned name space could be possible.

But perhaps our flight of fancy has gone too far into areas of DNS heresy with these thoughts. Even if we don't head into these areas of segmented flattened name spaces, there is still enormous potential value in DOH.

Why DOH?

There is some enormous value in the areas of enhanced privacy where the names being resolved are resolved without necessarily asking the public DNS resolution infrastructure, and DOH certainly can help in achieving this in certain situations.

DOH can simplify client systems. Crafting content delivery and name resolution into a single mechanism can certainly further the agenda of pushing the entire Internet through the eye of the TCP Port 443 needle. But it's more than that. If we consider environments that push complexity and functionality back into the server, and thereby permit simple transactional clients that have a bare minimum of functionality, then DOH is certainly useful.

There is value in pushing this information directly from the content server to the client application along a secured channel that is intended to be impervious to outside observation or manipulation.

There is value in using a TCP channel that does not have the issues with UDP fragmentation and truncation that we see in the DNS today.

DOH is certainly useful in all these areas.

But for me the real value of DOH lies in the simple observation that pre-provisioning is far faster than post-provisioning. In a delay sensitive environment there is considerable value in shaving delay off the DNS resolution times. A sure way to achieve this is by pre-provisioning the end user application with the resolution outcomes of names that the content server knows will be referenced in the delivered content. This ability to perform DNS name resolution as a HTTP push function embedded within content is what makes DOH worthy of attention above and beyond the privacy and robustness arguments that are largely shared by DOT and DOH.

Experimenting with DOH

The Mozilla Firefox browser has recently announced support for DOH. (https://blog.usejournal.com/getting-started-with-dns-over-https-on-firefox-e9b5fc865a43) This will allow the browser to use DOH in a number of modes, including running in parallel with conventional queries, running as a first preference for queries and using a conventional query as a fallback in the case of failure, or running DOH exclusively. DOH servers that can be configured with Firefox are available from a number of public DNS resolver services, including Cloudflare (https://developers.cloudflare.com/1.1.1.1/dns-over-https/wireformat/) and Google (https://dns.google.com).

Cloudflare have also released a DOH client (https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy/), which sets up a local DNS listener and passes all queries to Cloudflare's 1.1.1.1 DNS service over HTTPS as per the DOH specification. In this way all your local system's DNS queries and responses can be configured such that the external view is an encrypted packet stream using TCP port 443.

Further Reading

"Specification for DNS over Transport Layer Security (TLS)," Z. Hu, et. al., RFC 7858, May 2016. https://tools.ietf.org/html/rfc7858

"DNS Queries over HTTPS (DOH)," P. Hoffmann, P. McManus, work in progress, October 2018, https://tools.ietf.org/html/draft-ietf-doh-dns-over-https

The DNS Privacy Project Homepage, https://dnsprivacy.org/wiki/

Cloudflare: DNS over HTTPS https://developers.cloudflare.com/1.1.1.1/dns-over-https/

Google: DNS-over-HTTPS https://developers.google.com/speed/public-dns/docs/dns-over-https

"Hiding the DNS," November 2017. https://www.potaroo.net/ispcol/2017-11/2dns.html

"DNS Privacy", June 2016. https://www.potaroo.net/ispcol/2016-06/dprive.html

Postscript

The IETF has now published RFC 8484, "DNS Queries over HTTPS (DoH)" which is the complete specification for both DNS over an HTTPS transport session and the handling of the DNS as a HTTP web object. https://tools.ietf.org/html/rfc8484

Disclaimer

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

Author

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

www.potaroo.net