October 2018
Geoff Huston

Diving into the DNS

DNS OARC organizes two meetings a year. They are two-day meetings with a concentrated dose of DNS esoterica. Here's what I took away from the recent 29th meeting of OARC, held in Amsterdam in mid-October 2018.

Cloudflare's 1.1.1.1 service

Cloudflare have been running an open public DNS resolver service on 1.1.1.1 for more than six months now. The service, based on the Knot resolver engine, is certainly fast (http://bit.ly/2PBfoSh), and Cloudflare's attention to scale and speed is probably appreciated by users of this service. However, in a broader Internet environment that is now very aware of personal privacy it's not enough to be fast and accurate. You have to be clear that as an operator of a public DNS resolution service you may be privy to clients' activities and interests, and you need to clearly demonstrate that you are paying attention to privacy. Not only does this include appropriate undertakings regarding the way you handle and dispose of the logs of the resolvers' query streams, but also in being mindful of the way that users can pass queries to the service.

The 1.1.1.1 service supports DNS over TLS 1.3 and 1.2 using the GnuTLS system. This supports long-lived connection with client systems and also supports session resumption. There are a number of DNS tools that can use the DNS over a TLS connection including kdig (http://bit.ly/2A9f9rX, http://bit.ly/2OoLu7a), GetDNS (https://getdnsapi.net/), Unbound (http://bit.ly/2CIKEf0), Android P and the Tenta browser (https://tenta.com/).

The support for DNS over HTTPS for the 1.1.1.1 service is implemented using a NGINX front end that terminates the HTTPS connection and forwards the DNS payload to a LUA module on the resolver engine. It supports both the IETF's DOH wire format and a JSON encoding of the DNS as payloads in the DNS over HTTPS service. Firefox has recently announced its Trusted Recursive Resolver option to use DOH (https://mzl.la/2NGnQ0s) and Chrome is also working on this.

It has been common DNS folklore for many years that UDP is a more efficient transport protocol than TCP. The experience from these large public DNS resolvers is that there are tradeoffs here and holding a sessions state open across a number of DNS transactions allows for the session cost to be amortized over many transactions and avoids the issues with large responses causing retransmission and middle box interference. This applies to both the situation where clients are sending queries to a recursive resolver and where recursive resolvers are querying authoritative servers. This does not mean that the TCP load is equivalent to a UDP load, but the difference in server cost can be balanced against the enhanced integrity of the communications channel.

More generally, when looking at transport protocols for the DNS every option is a compromise. UDP has its issues with fragmentation handling for large responses, timeouts, open payloads and accessibility for on-the-wire intervention. It's also efficient and can be extremely fast. DNS over TCP port 53 has its problems, particularly with middleware. When APNIC Labs measured this a little while ago we found that some 17% of resolvers were unable to perform a query to an authoritative server using TCP, and this impacted some 6% of users. It appears that we have a lot of middleware that does not like packets using TCP port 53. TCP also requires a held session state on the server side, and this will reduce the peak sustained query rate in the server when using TCP transport

What about DNS over TLS? If we use the IETF-designated port 853 then the prospects for middleware interference look similarly frustrating. A distinguished port is a ready target for middleware filtering. Also, both TCP port 53 and TCP port 853 suffer from an exposed TCP control protocol that permits (cynics would go further and say "invites") middleware manipulation. TCP also suffers from head of line blocking, which can frustrate efforts to support multiple internal query channels.

If we want to improve this situation, we will need to obscure the TCP control protocol and these days that means that we are probably talking of DNS over QUIC. While QUIOC obscures the session control protocol from the network, and even permits internal multiple flow threading to circumvent head of line blocking, the question remains over the tolerance of middleware to allow UDP packets using port 443, Sadly, we can confidently expect some level of middleware damage with DNS over QUIC.

What's left?

We can do what the web already does.

DNS over port TCP 443, or DOH.

Cloudflare has introduced support for DNS over TLS and DNS over HTTPS for its 1.1.1.1 open resolver service. But it's DNS over HTTPS (DOH) that has attracted the most comment.

DoHysteria

Without doubt the topic of DOH has excited various passions in DNS circles.

When DNS transactions are embedded into encrypted into port 443 TCP streams much of the visible DNS is taken off the table. How big an issue is this?

It's clear that the current model of DNS resolution is, to significant extent, visible to a broader circle of onlookers. The upstream service provider typically uses DHCP to load your device with IP addresses, a gateway address and DNS resolvers. Supposedly it's a configuration that is visible to the user and can be overridden by manual configuration. But what happens when a browser uses its own configuration to select a DOH server? A concern from the "traditional" DNS folk is that this browser-based measure is an exception to conventional DNS provisioning process. It is probable that the choice of resolver service through DHCP and browser selection will be different. What will that mean?

While the DOH server is directly configurable in the recently launched Netscape implementation of DOH, it is also quite feasible for browsers to make their own decision about which resolver they should be using as a trusted recursive resolver over DOH This DOH mechanism can 'punch through' local provider-based infrastructure, and potentially locate the browser into a subtly different namespace. In today's environment where DNS responses are often tailored to steer the user to a particular service point, the prospect of having the browser hear different answers to the underlying platform's DNS environment can raise some interesting questions about naming consistency for users across applications.

Another concern of this browser-based model of DNS resolver selection is the risk that a browser will prefer to use a single open resolver service, leading to the potential for centralization of DNS resolution to a few providers. However, I'm not sure that use of DOH could significantly alter the current picture, where Google's public DNS service already provides a DNS resolution service to some 14% of the Internet's entire user population. If that's not centralization already, then I'm not sure what is! I'm also not sure that any concentration of DNS name resolution is necessarily the "bad thing" that others fear. Given the extent of local meddling and interference in many local DNS services, the existence of the option to use open public resolvers that have clear principles of honesty and privacy in their operation is perhaps one of the few Internet trust points left for many users! Sharing all your DNS transactions with Google may be preferable to the prospect of sharing these transactions with a large collection of unknown and unseen players who may not necessarily

have your interests at heart. And if you really don't trust what your DNS resolver is telling you then perhaps you may want to enable DNSSEC validation on the client side.

However, DOH will disrupt mechanisms that have implicitly assumed that the DNS service is an intrinsic part of the service infrastructure. DNS split-horizon hacks, that provide an "internal" name space as well as the public name space, will not necessarily work with DOH. On the other hand, such local DNS hacks never worked when users decided to use any of the open public resolvers. IPv6 transition technologies that relied on lying in the DNS, such as DNS64, may not work either, but the same comment applies that they may not have worked when users decided to use any of the open public resolvers.

I suspect that much of the reaction to DOH is that it takes the encrypted transport service for the DNS query resolution protocol and explicitly removes it from the IP service infrastructure and allows it to be used as an application level service. Applications can define their own trusted DNS resolution service, or even their own name space and do so independently of the platform upon which the application is running and also do so independently of the local IP service context in which the device itself is located. You could call this model of allowing applications the ability to determine their own trusted environment as one more instance of Internet fragmentation, and I'd agree with that characterization. However, it's not necessarily a bad thing.

The use of DOH by browsers may be a useful response for users when attempting to counter an environment of pervasive surveillance capitalism, where all parts of the Internet environment are susceptible to being coopted to diligently seek every last bit of information that is available about the activities of end users.

Changes in Versign-Operated TLD servers

UDP fragmentation is still a big issue for the DNS, particularly so for DNSSEC-signed domain names.

Verisign announced a change to its RSA key size for the TLD servers that it operates, and the key size will be lifted from 1024 bits to 1280 bits. That appears to be a rather arbitrary choice in a binary world. While RSA in theory permits any key size, conventional use has preferred key sizes that are a power of two, so when 1,024 bit keys are considered weak in today's environment, many operators turned to a key size of 2,048 bits, including the root zone ZSK.

But larger key sizes create larger response packets, and if you want to continue to use RSA and keep the signed NXDOMAIN response below an unfragmented UDP packets size of 1280 octets, then the decision to use a key size far smaller than 2,048 bits is a forced decision. It seems to be a palliative measure that addresses the issue that cryptographers consider 1,024 bit keys to be inadequate these days, but the additional 256 bits in a 1,280 bit key may not buy a large amount of time.

The options are to move to a 'denser' key using elliptical curve encryption, that impose higher compute loads on signing and verification but result in smaller keys for equivalent cryptographic 'strength', or live with a truncated response and push clients to use TCP to get the larger responses (as is currently the case with the root servers that Verisign operates). The other change is to remove cross-domain glue in the additional section of DNS responses. Again, this will result in smaller response sizes and pragmatically most resolvers do not trust cross-domain glue and will not use it in any case.

ECDSA in the root zone

When the root zone was signed back in 2010, the TLDs could also be signed. However, the management practices for the Root Zone (RZM) precluded the use of elliptical curve algorithms in the root zone's DS records for the TLDs.

In October 2017 the RZM practice was updated to admit the use of GOST R.34 10-2001, ECDSA P-256 SHA-256 and ECDSA P-384 SHA 384.

CZNIC, the operator of .CZ reported on their KSK algorithm roll from RSA to ECDSA P-256. The starting point was a 875Mb signed zone file with a DNSKEY response of 907 bytes. They then published ECDSA-generated signatures in the zone file, together with the ECDSA key in the DNSKEY RRSET, lifting the zone file to 1217Mb and the DNSKEY response to 1263 bytes. The removal of the old RSA keys and signatures shrunk the signed zone file to 695Mb and the DNSKEY response to 387bytes.

As already noted, ECDSA uses smaller signature, but takes more time to compete them and while the size of the signed zone and the size of individual signed responses decreased with the use of ECSA the zone signing time increased from 15 minutes to 22 minutes, which is consistent with expectations for ECDSA. The .BR TLD has also shifted to use ECDSA P-256 following the .CZ experience.

Hello-DNS

"As long as you are working on Hello-DNS you are not writing new DNS RFCs" PowerDNS's Bert Hubert pleading for a stop to adorning the DNS with further complex ornamentation.

An emerging theme within the DNS is the extent to which years of hacks and tweaks have changed the DNS into a formidably complex mess. it now is defined across more than 180 RFCs, and almost 3,000 pages of RFC documentation. The effort continues at an average pace of some two new pages per week (https://powerdns.org/dns-camel/). The collective achievements in the DNS are both impressive and frightening at the same time. Servers can deliver DNS responses at a sustained rate of up to one million responses per second, while at the same time we continually see the outcomes of complex and confusing specifications causing operational problems when DNS systems are embedded into high volume equipment. Little wonder that implementors make mistakes in the ever-increasing pile of specifications that define the way the DNS works today.

There have been a number of efforts to write even more RFCs that try and clarify the other RFCs. A recent offering in this space is the update to the terminology RFC7719 (the update is draft-ietf-dnsop-terminology-bis-07), that works at an even more basic level to provide clear meanings of the terms used in these specifications. Bert Hubert's position is that generating more RFCs will only make it worse!

Like many mature and ornate systems, the DNS is suffering from typical committee-induced bloatware. Individuals get rewarded in their efforts when they add even more bloat, but the rewards available to individuals for stripping down the system to its essentials are not so obvious.

Bert's response is perhaps unusual for the DNS community. Taking inspiration from Rich Steven's work in the Unix and Network Programming books "TCP/IP Illustrated" and "Unix Network Programming", he has embarked on a project to define a simple DNS authoritative server with both code and commentary in a form of "DNS Illustrated". The idea is an accessible, runnable entry point to the DNS which is minimal but complete. It's a teachable form of DNS in code and an associated commentary. In some ways this is a timely and useful commentary on where we are with DNS servers. What is the essential goal of all of this specification and software? What is at the heart of the DNS?

I like's Bert's pragmatic approach to addressing a real issue with the DNS today. For those of us who used the works of Rich Stevens and Doug Comer to guide us to an understanding of how the Internet was built through code examples, Bert's efforts might well help make the DNS more accessible. It may also help us to understand the distinction between the essential elements of the DNS and the extensive set of adornments to the core (https://powerdns.org/hello-dns/).

DNS Homographic Abuse

There are many ways to use the DNS in ways that are perverse, annoying or even fraudulent. At its stands the DNS is rather primitive in that "what you see is what you get". When you see a link with the label "https://www.potaroo.net" then it should take you to my site. what about "https://www.potaroo.net". The two DNS names may look very similar in some fonts, but to the DNS they are entirety different domain names.

This DNS name confusion is the result of a combination of two factors: the introduction of non-Latin scripts into the DNS and the use of Unicode to describe these non-Latin characters. We adopted the ascii character repertoire of the Latin script because each character glyph could be uniquely displayed, and we were all well trained in how to distinguish them from each other. For example, '1' and 'l' may look similar in some fonts, as do 'O' and '0', but we appear to be comfortable with this level of confusability. But with Unicode the character repertoire is an entirely different beast.

The aim of Unicode is not to define a collection of readily distinguished character glyphs. The aim is to allow an author to control what appears in the screen or on the printer. The fact that there are multiple ways to display the same glyph does not affect the functionality of Unicode. But whiule Unicode does not care, the DNS does. For example, I can generate a glyph that looks like a lower case 'a' using ascii code 61 (a) or Cyrillic code U+0430 (a). They look that same to me probably because my display system has chosen the same glyph to display them in this font, but the DNS represents these two characters entirely differently. There is a.example.com and xn--80a.example.com, and to the DNS they are certainly not the same domain name.

It's not surprising that folk have jumped on this as a way to mislead and deceive users, probably for profit. Passing off one DNS name for another can lead to all kinds of deliberately deceptive practices.

Can we stop it?

As Farsight's Mike Schiffman points out there is no silver bullet. The well-intentioned move to admit non-latin characters into the domain of discourse of the DNS has produced this chaotic and insecure environment and there is no easy way out. On the Internet we just operate on the principle that what we are seeing is what we are getting, even when it's not. There is no tamper-proof way of assuring a user that the server that they have accessed is the service that they intended to access. The 'real' and 'fake' domain names are probably certified to the same level and the appearance of the browser screen can be made to be identical. How is a user meant to distinguish between real and fake under those circumstances?

While it's tempting to declare Unicode a really bad idea for the DNS and just walk away from IDNs and non-latin scripts in the DNS, it's never going to happen. What can we do instead? Can a domain name owner protect the name by registering all potential homoglyph variants of the name? This seems unlikely. So it's a tradeoff that we all have to live with. We should clearly recognize that IDNs turn the DNS into an uncertain environment, and even the most prudent and careful of users may well be duped into entering their credentials into the wrong web site. And we just have no idea how we might mitigate this problem.

The Rolled Root Zone KSK

It happened. After ICANN decided to defer the proposed KSK roll in 2017, and after public consultation and much reflection and debate, and after many dire predictions of mayhem and disruption the root zone Key Signing Key (KSK) was rolled on the 11th October, and over the ensuing 48 hours the various caches in the DNS expired and the new key was used as the trust anchor for DNSSEC validation.

However, it must be recognised that ICANN undertook a significant campaign to inform service operators about this forthcoming roll and the apparent smoothness of this key roll may well be related to this significant energy expended in the planning process.

Now what?

There was some enthusiasm in the room to undertake this key roll on a more regular basis than every five years or so. There is much to be said in favor of an annual roll simply to prevent resolvers gluing themselves onto a particular trust anchor and wedging.

We also need to look at a number of further measures, including shifting the root zone key to ECDSA in order to fit all root zone responses more comfortably into unfragmented UDP. We also should grapple with the issue

of how to respond to various scenarios of that may call for an unplanned KSK roll. Can we get DNS resolvers to trust a "backup key" that we could roll into the root zone without an extended key introduction period?

It is also somewhat disturbing that the KSK roll was largely uninstrumented. There were two recent changes to assist in instrumentation, firstly trusted key signaling (RFC 8145) and secondly the key sentinel (not even an RFC was published in time!). Trusted key signaling caused us to defer the key roll for 12 months and frankly created a lingering sense of uncertainty over the readiness of the DNS to support the KSK roll. The key sentinel was introduced so late in the day that few resolvers supported the mechanism by the time of the key roll. In both cases what was clear was that we were seeing KSK trust signals from recently updated DNS resolvers, while our primary concern was old, crufty, unattended DNS resolvers that had turned on DNSSEC validation yet did not have any support for an automated trust transition (RFC 5011). These recent test mechanisms were simply unable to see further into the DNS environment and identify these potential points of concern. So a small counter-voice to the call to continue to roll regularly and do so more frequently than every five years is that we really are still in the dark here. It will be some time before we get widespread uptake of trusted key status reporting mechanisms and in the light of that observation we might want to be more deliberate about when and why we choose to repeat this KSK roll exercise.

DNS Aliasing

In this world of content hosting, DNS aliasing is incredibly useful. If I am hosting my server, located at "www.example.com" with a hosting entity that uses a common DNS suffic of "somecdn.corp" it is not unusual for the hosting arrangement to map the DNS name "www.example.com" as an alias of the hosting name "www.example.com.somecdn.corp". This arrangement allows the content host to steer the DNS request to the appropriate server address using the aliased domain name, while the user is still performing a connection to original name of "www.example.com".

And this is what the CNAME (Canonical name) record in the DNS is supposed to do. In the example.com zone I'd use the entry: "www.example.com. CNAME www.example.com.somecdn.corp." However, there is a limitation: If a CNAME resource record is present at a node, no other data can be present. That includes zone apex data such as SOA records, which implies that, in theory at any rate, CNAMES cannot apply an alias to a zone apex name. That's not the only limitation of CNAMEs. What if I want my web service to be hosted by one service provider and my mail to be provided by another? CNAME gets in the way here as the CNAME redirects all name types to the target canonical name for resolution. Also, what if I wanted to redirect the name queries at the zone apex? Strictly speaking, the DNS specification does not permit this, while many DNS resolution implementations allow it in practice.

Given this divergence between theory and practice there have been a number of alternatives proposed. It's not easy to tell if these alternative approaches improve the situation or merely escalate the confusion. DNAME defines a 'transition point' in the DNS that splices entire subtrees. But they work in slightly different ways, as the CNAME is an alias for the terminal label whereas the DNAME refers to a DNS tree splice at a particular label. If was want the label "color.example.com" to be equivalent to "colour.example.com" in all respects, including as a domain prefix, then we need to use both a CNAME and a DNAME as alias constructs.

ISC's Ondřej Surý spoke about his tests using this CNAME + DNAME construct. Surprisingly it looks viable, although there is still some distance between what can be made to work and what is wise to make work! As to the latter I think we are still unsure of how to handle various forms of aliases in the DNS.

Where to find more

These are just a few highlights of the DNS OARC 29 meeting. All the presentations can be found at http://bit.ly/2PahTxN

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