August 2022
Geoff Huston

DNS OARC38 Meeting

As I see it, the DNS is the last remaining piece of glue that binds the Internet together. We lost IP address coherency within the Internet many years ago and the DNS is all that's left. Consequently, the DNS is vital for the Internet. Perhaps the most critical question to emphasise the central role of the DNS is the question: "Would the Internet still function if the DNS were to just cease working in the next minute or two? " The answer is undoubtedly "No!"

Yet there is still much in the way the DNS behaves that we really don't know, much we would like to do that we can't do already, and much we probably want to do better. Meetings of the DNS Operations Analysis and Research Centre (DNS OARC) bring together a collection of people interested in all aspects of the DNS, from its design through to all aspects of its operation, and the presentations and discussions at OARC meetings touch upon the current hot topics in the DNS today. OARC 38 was held in Philadelphia at the end of July 2022 and was part of the slow return to a post-covid world, where some of the participation was in-person and some online. Here's my impressions of some of these presentations from the meeting.

DNS over QUIC at AdGard

AdGard presented on their experiences in DNS over QUIC. AdGard is a relatively recent entrant in the open public resolver landscape, with a public service launched in December 2018. As the name suggests, the intent was to provide a DNS resolution service that blocked the presentation of Ads by using DNS filtering. They've been one of the first open resolvers to support access via DNS over QUIC.

> QUIC is essentially a transport protocol that combines TCP and TLS into a single protocol that supports stream multiplexing, using a basic UDP presentation to the network and cloaking all transport controls and the payload behind an encryption veil. It also removes one round trip time from the handshake process, allowing for faster session setup.

Being an overlay DNS resolver, the user base of AdGard are clients who have deliberately chosen to use this service, and they're not using the conventional default settings of their service provider. It's challenging to ascribe motives to this choice, but there are a few reasons why a client would elect to use AdGard's services via an encrypted DNS session. Not only is the user not exposing their DNS activity to the local environment, but by using an encrypted tunnel it bypasses any local measures that may intercept and potentially manipulate DNS responses.

AdGard report that some 14% of their DNS traffic is over UDP port 53, 70% uses DNS over TLS, 15% using DNS over HTTPS and 1% using DNS over QUIC. Considering that implementations of DNS over QUIC are very recent, even 1% of traffic is surprising at this point.

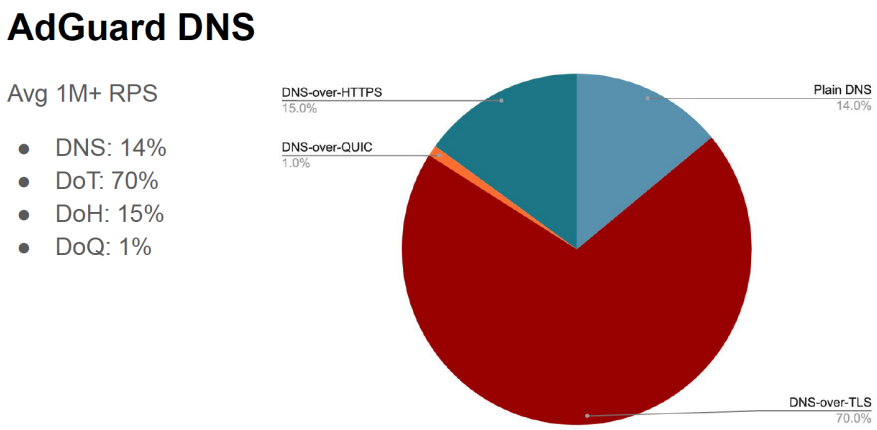


Figure 1 - DNS Transports as seen by AdGard, from "DNS-over-QUIC", Andrey Meshkov, Adgard

AdGard report that DoQ connections are more stable than DoH or DoT, in that the connection is held up for an average of 30 queries, while DoT connections are held for an average of 9 queries and DoH for an average of 14 queries. They report a server CPU load that is commensurate with DoH, which is slightly higher than DoT, with some 60% of the server CPU time spent in QUIC code. (Although it is unclear to me in this if the kernel TCP code of DoT and DoH was included in this comparison. QUIC is implemented as a user level module.

DNS over QUIC for Authoritative Servers

The work on the IETF that generated standard specifications for DNS resolution over encrypted channels for the stub-to-recursive resolver scenario has largely completed with the work of DoT, DoH and DoQ. Attention has shifted to the potential application of these same techniques for the scenario of the recursive resolver-to-authoritative server. The issue here is that a recursive will not necessarily maintain a long-held session with a single authoritative service, so the ability to amortise the cost of the initial session setup over a large set of subsequent queries is not immediately apparent in the recursive-to-authoritative scenario. However, there are two authoritative services that are susceptible to DoQ, namely AXFR and IXFR. This was examined in a presentation by CZ.NIC.

DNS over UDP represents a highly efficient option, and with an XDP implementation of UDP and individual servers can be dimensioned to answer some 10M queries per second. However while they are highly efficient, they are also susceptible to various forms of UDP DOS attacks, and there is the issue of IP fragmentation management and TCP signalling for larger responses. A TCP-based service is not susceptible to spoofed source addresses, and the managed stream does not have the same IP fragmentation issues, which there is still the issue of SYN attacks and Slowloris attacks over TCP. An XDP-based TCP service can be dimensioned to process some 1M queries per second. DoQ has roughly the same vulnerability profiles as TCP. There is an option to use XDP to implement QUIC. Current implementations are heavy in CPU and memory use and the connection capacity is reported to be considerably lower, at some 10,000 connections per second. QUIC can use the retry packet in response to initial connections to ensure that the source address of the client is not spoofed, at the cost of an additional RTT.

The folk at CZ.NIC have been experimenting with the KNOT server in conjunction with the Libngtcp2 and Gnutls libraries. The benefits of this approach are a lower latency than TLS for an encrypted channel, at the cost of a slightly greater CPU and memory load. It's still early days for QUIC and its likely that we will see improvements in this picture in the light of further operational experience.

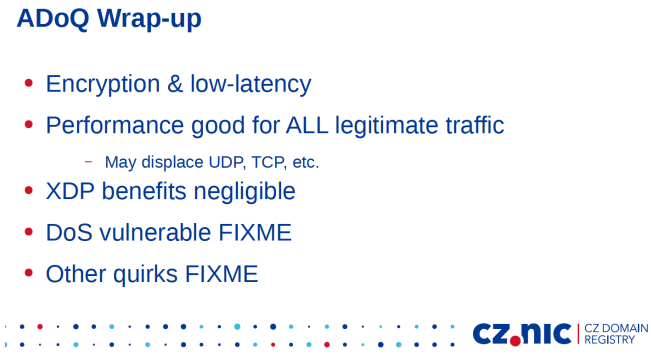


Figure 2 – ADoQ Summary from DoQ in Authoritative by Libor Peltan, CZ.NIC

DNS Spoofing Countermeasures at Google

In a world where every DNS name was DNSSEC-signed and every DNS client validated all received DNS responses, then we wouldn't necessarily have this problem, nor would the problem be as general of all DNS transactions used encrypted channels. However, while the overwhelming majority of DNS names are not signed, and most of the DNS transactions use open channels, it's still possible to inject false answers into the DNS and thereby deceive unwitting end clients. One could perform this injection close to the edge of the network and impact stub resolvers if you wanted to mount relatively precisely targeted attacks. A broader attack can be mounted by passing falsified responses to recursive resolvers, and in this respect Google's Public DNS service represents an attractive target.

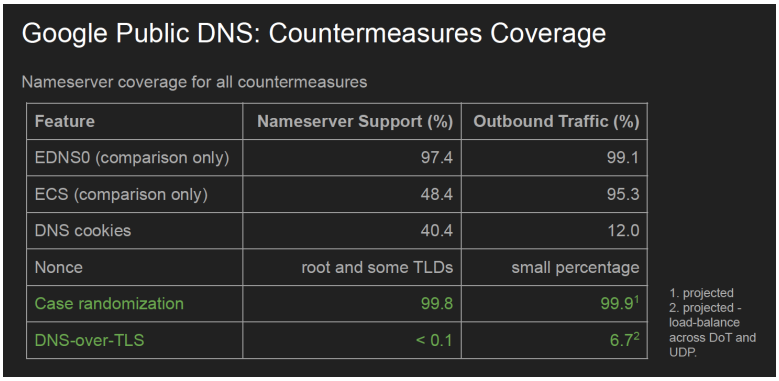
RFC3833 (Threat Analysis of the DNS) described potential attack vectors, including packet interception by on-path attackers, and query prediction and efforts to inject a response, as described in RFC5452 (Measures for Making DNS More Resilient against Forged Answers).

There is little a DNS recursive resolver can do against on-path attacks if the name is not signed and there is no way to validate the response other than use Authoritative DNS over TLS or QUIC whenever possible. However, for such a direct substitution attack to be effective, the attacker has to place itself on the response path in the first place. It is possible for an attacker to force a target resolver to pose a DNS query and then attempt to forge a first response directed to the resolver that matches the original DNS query and query ID, matches the address and port fields. Guessing the query ID is feasible if the attack is persistent. RFC5452 suggests that a resolver should use random query ID and varying (and random) source port values on its queries. More recently, RFC7873 and RFC 9018 propose DNS Cookies as a measure to counter IP address spoofing. Authoritative DNS over TLS or QUIC would also help here, as spurious responses can be readily rejected by the resolver.

Google has implemented the RFCb5452 countermeasures and DNS Cookies as part of its measures to counter such attacks, but their measurement of name server capabilities show that these measures are insufficient. Primarily the problem appears to be non-compliant authoritative servers returning incorrect responses for DNS Cookies. Google have been performing a structured regular measurement to ascertain the level of DNS protocol compliance performing a daily query of the top 1M nameservers (as per Google's ranking). Their findings from this study are that DNS Cookies are only supported in 40% of these name servers, and only supported in 12% of the query traffic.

Consequently they apply a number of mesures that have been proposed in the past. One is based on draft-vixie-dnsext-dns0x20-00, which proposed randomisation of the character case in the query label. The attacker has to match the case of the query name used in the unseen query. Of course, this randomisation is most effective in longer query names. The good news is that case integrity where the case of the query name is reproduced in the query section of the response is supported by some 99.8% of all surveyed authoritative nameservers, so this is a useful additional to the approaches to minimise the risk of cache poisoning s in off-path attacks.

Another measure uses a prepend of a nonce to the queries to root and tld labels, and once more the attacker has no direct knowledge of the value of the nonce in the query, again making a flooding guessing approach less likely. According to Google's note on this approach (https://developers.google.com/speed/public-dns/docs/security?hl=en#nonce_prefixes): "Although in practice such requests make up less than 3% of outgoing requests, assuming normal traffic (since most queries can be answered directly from the cache or by a single query), these are precisely the types of requests that an attacker tries to force a resolver to issue. Therefore, this technique can be very effective at preventing Kaminsky-style exploits." The exceptions that break the universal application of this approach are when the TLD nameserver is also authoritative for other second level names, such as a server being authoritative for both .uk and nic.uk, and certain forms of glue records. Google address these issues with an exception list of TLDs where the nonce query is not effective.



Google Public DNS: Countermeasures Coverage

Nameserver coverage for all countermeasures

Feature	Nameserver Support (%)	Outbound Traffic (%)
EDNS0 (comparison only)	97.4	99.1
ECS (comparison only)	48.4	95.3
DNS cookies	40.4	12.0
Nonce	root and some TLDs	small percentage
Case randomization	99.8	99.9[1]
DNS-over-TLS	< 0.1	6.7[2]

1. projected
2. projected - load-balance across DoT and UDP.

Figure 3 – Anti-Spoofing Effectiveness measures from Cache Poisoning Protection foe Authoritative Queries, Puneet Sood, Google

Other measures used by Google include randomising the choice of name servers. stripping duplicate queries from the outbound queues and performing rate limiting.

Google also perform unilateral probing for support of Authoritative DoT support . To avoid some of the incipient incremental load issues for DoT in CPU and memory they load balance across Do53 and DoT transports. Such TLS queries are far more challenging for an attacker, and the hope here is that the cost of any form of successful attack on DoT is prohibitively high. AS Google's survey notes, the level of authoritative support for DoT for queries is less than 0.1% of servers, but on a more encouraging note, this corresponds to some 7% of query traffic.

Google projects that these spoof protection measurements will cover 99% of queries after the various rollouts are complete. The use of Authoritative DoT (ADoT) will account for between 2.5% and 6.5% of queries, query name case randomisation in UDP accounts for 42% of queries and is anticipated to rise to 90%. Current use of DNS Cookies is at 0.,1%, although this too is anticipated to rise to around 10% with cookie auto-detection. The nonce query is a specialised case set of queries at the top to the DNS name hierarchy and covers around 0.75% of queries. The intention is to concentrate their efforts on unilateral probing to support queries over ADoT, as the experience to date has been positive. They also intended to increase the scope of the use of DNS Cookies with auto-detection and enable case randomisation by default.

The presentation included a useful guide to operators of authoritative name server infrastructure to assist in supporting standard anti-spoofing countermeasures:

- RFC 7873: DNS cookies by upgrading to recent name server software with support, or adding support to your server. It is also appropriate to support RFC 9018 Interoperable Domain Name System (DNS) Server Cookies. A side benefit DNS Cookies can verify the validity of a client's IP address
- Follow RFC 8906 [BCP 231] recommendations on responding to queries
- If you cannot implement DNS cookies, ensure case for query name in response is preserved

- Experiment with DNS-over-TLS if you have the option DoT (and DoQ) avoid issues with UDP queries while also providing some privacy benefits as well.

Most of this material can be found in a writeup by Google at https://developers.google.com/speed/public-dns/docs/security. Even so, I feel I must make the observation once more: In a hypothetical world where every DNS name is DNSSEC-signed and every DNS client validates all received DNS responses, then we wouldn't necessarily have this problem!

The Cost of DNSSEC Validation

The protracted progress of adoption of DNSSEC continues to be a vexatious issue. Surely, having a DNS system that allowed greater levels of trust in the currency and authenticity of DNS information would be a good thing? So why are we so reluctant to all pile into DNSSEC?

There are many answers to this question, but in an Internet that is still highly motivated to make it all faster, one common theme is that DNSSEC validation is just too slow. The story is that we don't sign DNS names because too few users validate DNS responses, and we don't validate DNS responses because validation it's too hard, too slow and too expensive.

Answering the first concern, that validation is too hard to set up is readily addresses. In the BIND named daemon the configuration commend to enable DNSSEC validation enabled in a single one line entry: `dnssec-validation auto;.`

What about validation being too slow and too expensive? As a paper exercise, DNSSEC validation has a considerable additional overhead. For each zone in the delegation hierarchy of a DNS name the validator must retrieve the signed DS and DNSKEY resource records, and then construct a validation chain between the signature being validated through the DNSKEY and DS records to the Key Signing Key of the root zone. That's additional work in terms of queries, elapsed time to complete name resolution, increased network traffic, and increased processing and memory loads. But that's what a paper exercise might lead you to believe. But the entire process of name discovery in the DNS through query iteration would lead the same paper exercise to believe that the entire DNS in infeasible. Yet it works, and surprisingly efficiently. So is DNSSEC validation in a similar state, where the concerns about the additional overheads are overstated?

To try and answer this question ISC's Petr Špaček used a DNS query log from a European telco from February 2022 and replayed it through a recursive resolver comparing the response times from the resolver with DNSSEC validation enabled and with it disabled.

The first test used a replay rate that resulted in 10 minutes of queries with an average query rate of 9K queries per second. In the initial 60 seconds, the cache was quickly populated and more than 91% of queries were answered within 1ms irrespective of whether DNSSEC validation was enable in the resolver. The other 9% of queries took up to 1 second to answer, but the distribution of response times was similar when DNSSEC validation was enabled or not. Looking at the final 60 seconds of data, when the cache was populated some 96% of queries were answered within 1ms using the local cache, irrespective of whether DNSSEC validation was enabled or not. And the profile of response times for the other 4% of queries was tmuch he same irrespective of whether validation was enabled, indicated that the cache was already holding the DNSSEC key records so little time was spent in performing key fetches.

The profile of TCP socket use was very similar with some 61 active TCP sockets across the last minute of replay for both validating and non-validating scenarios. Similarly, the query traffic volumes were largely the same once the cache warmed up. Perhaps less obviously, the CPU loads where also very similar after the first 30 seconds, with the validating scenario loading the server to an average of 57% as compared to an average of 55% for the non-validating case. The only case where there was an appreciable change was in the case of validation where the memory requirement was some 10% higher, presumably because of the cache requirements for the DNSSEC resource records.

The exercise was repeated using a significantly higher query replay rate, sending an average of 135K queries per second. In this case the response times in a cold cache in the first 60 seconds were appreciably higher for the validating scenario, increasing response latency by a little under a factor of 10 for the 25% of cases where the response latencies differed. One the cache was warm there was no appreciable difference.

The answers from this exercise are that today a recursive resolver could enable DNSSEC validation and experience no appreciable impact on latency, bandwidth, CPU consumption and I/O socket use. Memory requirements would increase by around 10%, but that would be pretty much the extent of the impacts of this change.

I suppose the major takeaway for me from this presentation is that the underlying enabling factor for the DNS at large is that caching works exceptionally well. It appears that the profile of users' DNS queries has a very high level of self-similarity and in such a situation a local cache is able to respond to almost all queries directly. Not only does this apply to the DNS in general, it also apples to DNSSEC validation as well. The additional information required to validate DNS responses is also readily stored in the local cache, and of course once the cached information is validated, it can be served directly as validated data without any extraneous re-computation of the validation chain.

Improving Root Service

The system of servers that serve as authoritative servers for the root zone of the DNS has changed is so many ways over the years. In an earlier era each root server operator operated a single hardware system in a single location. The number of distinct servers was a tradeoff between resiliency and performance, where more servers in more locations was intended to provide a better service and the protocol limitations where packing a DNS response with an arbitrarily long list of root name server names and addresses exceeded the DNS protocol's capabilities.

The adoption of anycast for DNS nameservers took some time to be accepted, but it transformed the nature of the root service and performance objectives of the service as well. The current census of root service instances has 1,533 servers operated across the 12 root server operators (https://root-servers.org/). It's likely that this expansion of the root service via these anycast constellations will continue, but there is an underlying question each time when the anycast constellation is expanded as to whether the additional server is making an overall improved service, for how many end clients, and by how much.

Now a naïve approach to this question would be to take a sample set of queries seen at root servers, perform a ping and traceroute of these addresses from the root service location(s), calculate the mean and variance of network latencies and response times and derive benchmarks of median service levels. These levels can then be used to identify areas where the existing anycast constellations provide substantially poorer service outcomes, and this data could be used to motivate further expansion of the anycast service constellations. The glaring assumption here is that the raw query log data of the root server(s) is available for analysis, and the issue here is that the privacy issues around the DNS and logs are sufficient to make this a tough assumption these days.

So how can one estimate the efficacy and performance of any any service constellation when the essential data relating to network location and latency is deliberately obscured?

There are some deeper questions going on here. Are the details of anycast distracting us from the more important questions about the evolution of the root service? If the profile of answers to root server queries is much the same as it was from some years back, where the overwhelming majority of answers was "no such name" or the NXDOMAIN error code, then what is being achieved by saying "No" faster? Also, given the relatively small size of the root zone is it still optimal for recursive resolvers to incrementally discover the content of the root zone by incremental queries, or does it make more sense

to deliver the entire zone to the resolver through some transfer mechanism? Incremental queries made some sense in a kilobit network but seem to be just entirely inappropriate in a gigabit network. Should we put our effort into an ever-expanding anycast service network to cope with more incremental queries, or look at approaches such as "hyperlocal" (RFC 8806) to completely redefine the relationship between the recursive resolver set and the root zones? Should we be looking at the zone as a single digital artifact and transform the nature of its distribution from piecemeal on-demand incremental pull via queries to pre-provisioning via whole-of-zone push?

Zone Message Digest for the Root Zone

Which brings me conveniently to the presentation by Duane Wessels of Verisign on ZONE MD, or the inclusion of message digests in DNS zones, and most particularly, for the root zone. A ZONe MD record is intended to contain the cryptographic digest, or hash, of the data in a DNS zone, and it is embedded in the zone data itself as ZONEMD resource record. It is computed by the zone publisher, and verifies by recipients of the zone (RFC 8796). Ideally the ZONE MD record, like the zone itself, is DNSSEC-signed, so any efforts to tamper with the zone contents are immediately apparent even if the tamperer attempts to substitute a new ZONE MD value.

ZONE MD is making its way through DNS implementations with the capability to generate ZONE MD records implemented in a number of DNS implementations (ldns, dns-tools and Knot DNS) and the capability to verify a ZONE MD value in a larger set of implementations including the listed publishers and also Unbound, NSD and PowerDNS Recursor.

Making changes to the root zone is a ponderous, involving much in the way of consultation and confirmation. The Root Zone Evolution Review Committee (RZERC) of ICANN has recommended that ICANN oversee a plan to add ZONE MD records into the root zone. This is now underway and while it won't happen overnight there is a clear intention to see this through.

For those so motivated, anyone can find ZONEMD-enabled root zones for testing at http://zonemd-testing.verisignlabs.com/.

DNSSEC and NSEC3

One of the major challenges with the design of DNSSEC was to device a design that allowed the absence of information to be validated. This applies equally to RR Types of an existing name and the existence of the name itself. The degree of difficulty was compounded by a desire not to introduce a dependency on on-the-fly dynamic signing, so the idea was to be able to generate proof-of-non-existence records without knowing in advance what the queries will be. Responses of the form "*the name you have queried, foo, does not exist*" is not helpful in this scenario. The alternative was to generate an ordered list of the names in a zone, and pre-sign all the "gaps" in the zone. This would mean that these pre-computed responses would look like "*all names between bar and foo do not exist in this zone*". This is fine if the zone publisher is happy to disclose the entire contents of the zone, but less fine if the zone publisher is unwilling to do so. From this came the concept of NSEC3, where the zone's labels were hashed and the NSEC records were based on the ordered list of hashed zone values. When given a response of the form "*all names whose hashed value lies between hash value 1 and hash value 2 do not exist in this zone*" the verifier needs to hash the queried name and confirm that it lies between the two hash values. They will not be able to reverse the hash and determine the original label values.

However, hash functions are not all that opaque and there are a number of tools out there that can do a decent job of reversing a hash function and listing the candidate original values. In an effort to make the hash function more opaque, the specification added both salt and an iteration count. The salt value was added to the initial value, and the hash function was re-applied to the hash value a number of times (the iteration count).

But it turns out that neither of these operations changes the opacity of the hash function at all, and a salt of zero and an extra iteration count of 0 is functionally equivalent to any other combination of salt values and iteration counts. High iteration counts are a case on imposing load on the publisher and the verifier but not on the attacker! The new recommendation is to insist on an empty salt value and an extra iteration count of 0.

What should a validating recursive resolver do when it encounters a zone with a non-zero iteration count? The draft recommendation proposes that the response be marked as insecure or return the SERVFAIL response code. The issue here is that this may result in otherwise valid but unresolvable names. A survey by JPRS has found some 1,149 TLDs that use NSEC3 records with non-zero iteration counts, so a default action to treat such zones as being unresolvable is perhaps not what was really intended here.

The Cost of ADoT at a Root Server

There has been much in the way of recent activity to develop a model of recursive resolvers querying authoritative servers using DoT and its brethren (DoQ and DoH). They have a higher cost of session establishment, and an incremental cost associated with the crypto processing component, but the ability to operate sessions across multiple query/response transactions can amortise the session startup cost across the subsequent transactions such that the average cost of the encrypted transport options compared to UDP is just the crypto component. However, this is not necessarily the case for the scenario of recursive resolvers querying authoritative name servers.

The team looking after B-Root at USC/ISI and Google performed an interesting experiment by sending some 40%-50% of traffic between Google's recursive resolver in Singapore and a B-Root instance in the same geo location over TLS. The first week was without TLS enabled, and the second was with TLS enabled. In this case they were looking at the profile of packet load, bandwidth and root server load.

Their findings are informative. When the recursive resolver uses ADot half of the time the received packet count rises by a factor of 2.12 and the transmitted packet count by a factor of 1.54. The bandwidth requirements rose in a similar way, by 1.9 on received traffic and 1.6 on sent traffic. The processing requirements rose by a factor of 1.6.

Their conclusion is that for a root server talking to a relatively busy recursive resolver the increases in packet load, bandwidth and processing are not outlandish, and from this perspective ADoT in this scenario is feasible.

However why would you want to do this between a recursive resolver and a root server? If the answer is to prevent spoofing attacks then there is some considerably merit here. the root zone's delegation (NS) records are not signed in the root zone, so DNSSEC validation would not necessarily detect spoofing, while a TLS tunnel would render third party spoofing extremely challenging to the point of practical impossibility.

It's true that the incremental cost is not negligible, and the higher setup cost cannot be completely amortised over subsequent queries in ADoT. The judgement call is whether the value of these measures as an anti-spoofing measure is worth the additional cost in server (and resolver) load.

Vendor Product Roadmap

I have found these sessions to be highly informative in past OARC meetings. What are the DNS software vendors currently working on?

In the case of the NSD server from NLnet Labs there is work underway in supporting Extended DNS Errors, SVCB/HTTPS RR Types, interoperable DNS Cookies and DNS zone transfers over TLS. The current plans include redesigning zone parsing and loading, use of an adaptive radix tree for the internal

data store, more XDP support and Catalog Zones support,. They are also working on DoQ and unilateral probing for ADoT/ADoQ.

For the unbound resolver there is more work on resiliency, including supporting serve stale, and reworking the timeout and retry strategies. They have added support for ZONEMD, SVCB/HTTPS RR Types, TCP/TLS stream reuse and Extended DNS Errors. The planned work includes client side DoQ support, client side proxy v2, and upstream DNS Cookies.

The KNOT server has been working XDP-TCP and XDP-QUIC implementations. They have also been improving support for AXFR/IXFR when the server is serving a large number of zones, and Catalog Zones support. The KNOT resolver has extended DNS Error support, ZONE MD and Proxy v2 support.

The PowerDNS recursor has added support for aggressive NSEC cache, EDNS0 padding, DNSSEC validation enabled by default, Extended DNS Errors, and outgoing TCP Fast Open. They have support for manually configured ADoT and unilateral DoT probing and ZONE MD. Thbeir dnsdist product has support for PROXY v2, outgoing DoT and DoH, SVCB/HTTPS and TC responses via XDP.

BIND is moving to a different internal data structure intended to support faster load times and less memory footprint by using a qp-trie structure. They are also supporting Extended Errors, catalog zones and stream DNS for both TCP and TLS transports.

DNS OARC 38

The presentations from the meeting are available at https://indico.dns-oarc.net/event/43/. The next OARC meeting is scheduled for October 2022, located at the same venue as RIPE 85 in Belgrade, Serbia on Saturday 22nd & and Sunday 23rd October.

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