November 2021
Geoff Huston

# DNS at IETF 112

Virtual meetings continue in the IETF, and the latest one was IETF 112 in November. Here are notes from some selected working group meetings that caught my attention. And, yes, I should say at the outset that the DNS continues to catch a lot of my attention these days, so I'll divide this report into DNS and the other topics. This is the DNS part.

## DNS Operations - DNSOP

DNSOP held two sessions in IETF 112, and I'll run through the major topics discussed there.

### I'll take my NSEC3 unsalted please

The concept of authenticated non-existence of a domain name in DNSSEC and obscuring the complete contents of the zone at the same time, which is what NSEC3 is attempting to achieve, continues to confuse us.

NSEC3 was a late addition to the DNSSEC design, and it tried to achieve three objectives: Provide authenticated proof of non-existence of a domain name, prevent zone enumeration, and permit opt-out such that blocks of unsigned delegations could be covered by a single NSEC3 record.

The operation of NSEC3 is simple — the names are obfuscated by hashing them using a SHA-1 hash, and it's these hashed names that form the basis of the ranges used to 'stride' the gaps in the lexically ordered enumeration in the zone's name set. In other words, rather than an ordered list of names in a zone using conventional lexical order, NSEC3 uses the hash of these names as the ordered list of names in a zone.

NSEC3 records have several parameters that are specified via an NSEC3PARAM record at the zone apex. This record specifies the hash algorithm, processing flags, the number of hash iterations and a salt value. Although Section 10.3 of RFC 5155 specifies upper bounds for the number of hash iterations to use, there is no published guidance for zone owners about a good value to select. Obviously, hashing the hash value will produce an even better hash. Right? Well, no. Hashing provides only moderate protection, and it is now understood that further iterations of a hash function only serve to add further processing load without any incremental level of protection. It's a classic case of making both the defender's job and the normal user's task harder without making any difference to the attacker. Rehashing a hash still produces a hash. Nothing really changes.

In the light of this understanding, it seems only pragmatic to recommend a salt iteration count of 0. Salt was meant to make dictionary attacks harder, but as the hash is computed across the fully qualified domain name in any case, dictionaries don't work across zones and there is no additional protection provided by a salt value. This parameter can also be used as a compute attack on validating resolvers by making them perform additional work to validate the NSEC3 responses.

So don't use it.

It's simple to say, but these days DNS engineering has a massive component of primitive superstition and ingrained habit, and there is some aversion to heeding a simple message of 'stop this silly practice!' The current version of the draft allows for validating resolvers to treat the zone as insecure if the iteration count is more than some value. This option is not universally favoured in the working group and there is a sentiment not to downgrade the secured state of the zone and for the validator to instead just return SERVFAIL for any salt iteration count greater than some value.

If opt-out really is the objective, then NSEC3 still has some value for some, I guess. As a means of obscuring the contents of the zoner it's a poor technique, as the hash function is readily broken with a decent graphics engine and any one of a number of published tools. If a zone admin is strongly motivated to counter zone enumeration, then there are other approaches, including the 'white lies' approach described in RFC 4470 and RFC 4471.

## DNSSEC automation

Like any other aspect of Internet services there is an increasing desire to improve the resilience of the DNS, and DNSSEC in particular. This is often implemented using multiple providers of the service where, in theory, operational incidents that may impact one provider would not impact all the other providers at the same time.

Signed zones should be no different to unsigned zones in terms of multiple providers, and they are if a pre-signed zone is passed to each provider to serve. However, as zones get larger and as responses are customised then there is a temptation to place the signer function into the responder, signing these responses on the fly. This has its problems with multiple providers, particularly when you would like to avoid messy processes to allow these providers to share a single zone signing key. The alternative is to allow each provider to use their own zone signing key and let them sign responses with their own key. This does not obviate the need for the multiple providers to inform each other of their respective ZSK, as the DNSKEY resource record is the one record where all the ZSKs need to be included. There are no tools out there to assist in automating this process of synchronising across such DNS providers.

There are some related issues in this space, including the bootstrapping problem of transitioning a zone from unsigned to signed as it relates to the view from the parent zone. While various polling tools, such as CDS, are useful once a secure signed delegation has been set up, getting the initial state in place while avoiding the trust on first use gap remains an issue.

## Domain verification

Getting a certificate from a CA through automated DNS-based domain verification has been an ad hoc affair with each CA validator adopting its own challenge/response mechanism. There are a different set of strengths and weaknesses in these approaches and often the validator adopts a particular validation process to meet their particular requirements. (Try `dig TXT bbc.com` for a good example of the result of this diverse set of domain validation approaches.)

This appears to be an area that could benefit from further study, particularly in terms of the analysis of the various ways in which these systems can be attacked and subsequently to specify domain verification procedures that are robust and could be adopted by all. After all, isn't this part of the role of technology standardisation? Or doesn't the IETF do that these days?

## Structured data for 'DNS access denied' error page

The DNS has been a somewhat terse protocol, particularly when things are not going as expected.

This work is proposing to add a commentary to a filtered DNS service which sends back an explanation as to why the query was filtered. There are some odd nuances here, particularly where a DNS resolver can intrude upon a DNS resolution request and place a message into the transaction that is passed back to the user.

This proposal appears to be some form of addendum to the extended error code work, but the ability for DNS middleware to convey messages to the end user raises some challenging questions.

One of the more fundamental questions concerns the design of the DNS resolution protocol, and the deliberate decision for the interactions within the DNS infrastructure to be entirely opaque to the original querier. So why would an error message from some unknown source reporting some DNS error have any context and meaning to a user? When I look at the DNS queries my device emits, then I guess I can recognise only a tiny fraction of them as names I provided. The rest relate to the inner workings of these applications and the device and an error message about a query I never knew I made seems entirely random to me!

I suspect that the deeper problem is this opacity in the operation of the DNS resolution protocol and the associated lack of diagnostics, and I suspect that opacity was a deliberate design decision at the time. I'm not sure that these efforts to retrospectively attach such a diagnostic function into the DNS is going to be a helpful change to the DNS environment. I'm more aligned to the thought that it isn't.

## DNS Catalog Zones

This concerns the automation of zone provisioning between the zone administrator, and primary and secondary name servers. The capabilities include change of ownership of the zone, grouping a set of zones with particular properties, and serial number control.

All this points to emerging needs of role specialisation and increasing centrality in the DNS infrastructure world and the need to automate the administrative actions associated with the operation of a zone across the various providers (primary and secondary servers).

I find it odd to hear the IETF decry this centrality as being harmful to the dispersed and decentralised model of the Internet, yet at the same time to see us quietly working on standard approaches to address the technology needs of centralized service orchestration. I guess we forgot how damaging it was for a standards body to have religious opinions, and the last time we tried this opinionated stance was in the collective refusal to standardize NATs and that didn't stop NATs. It just turned NAT traversal into a convoluted guessing game! If we need tools to orchestrate name service provisioning, then let's produce standards to allow these tools to interoperate well.

## Automated DNSSEC Bootstrapping

These days the DNS infrastructure operation includes a complex dance between the DNS Provider, a Registrant, a Registrar, and the Registry operator. That means when anyone wants to add to the information transferred between these actors then a lot of provisioning tools must be altered and the resistance to change can be quite high.

Some years ago, the concept of CDS/CDNSKEY records were introduced, allowing a parent zone administrator to poll the delegated child zone directly. If the CDS/CDNSKEY records have changed then the new values are pulled from the child zone, validated using the zone's old keys, and then used to replace the DS entry in the parent zone. This is all good if the zone is already DNSSEC-signed, and a DS record already exists in the parent zone.

Options are being explored to figure out how to bootstrap this DS record in a secure manner. The approach described to the working group in this session was to bootstrap the CDS/CDNSKEY records using a secured zone that already has trust established with both the parent and the delegated child. The initial records are published both at the zone apex and in this trusted location, and once the records are checked against the zone apex, the trust bootstrap record can be removed, and the client zone is now in control.

## Authenticated ADoT

This is a measure to encrypt the next part of the DNS infrastructure, namely the path between the recursive resolver and the authoritative name server. The issue is that there are a number of parts to proposals to address this, and these days we have a number of working groups, including DNSOPS, DPRIVE, ADD, and now DANCE. Some activities don't neatly fall into one working group or another and this activity, using TLS to encrypt the DNS transactions between a recursive resolver and authoritative nameservers, referred to as Authoritative DNS over TLS (ADoT), is one of them

There are several choice points here and they vary between just using TLS to protect the network path through encryption without authenticating the identity of either the recursive resolver or the authoritative server through to authentication of the server to the other end of mutual authentication of client and server.

Looking at the middle case of authentication of the server, or Authenticated ADoT, the problem is that the delegation NS record needs to be protected before its used (as distinct from the conventional DNSSEC validation function that only validates the response, not the delegation path that was used to get to the response). In DNSSEC's design, the parent's NS records are not signed, and in ADOT this is a point of vulnerability. These parent-served name server names need to be DNSSEC-signed in order to convey the name to the client in a robust manner that detects attempts to tamper with the name. When the client initiates a TLS connection, the server at that IP address needs to present a TLS domain name certificate that matches the name, and the client needs to validate that name.

This could be performed using a conventional certificate validated through the WebPKI, but to try and contain the secure space with the DNS and not introduce more external dependencies the alternative approach to validating this name can be based on DANE (Domain Keys in the DNS) and use a TLSA record for the nameserver name. In addition, it is proposed to provide explicit signalling for ADoT support, rather than use an extended process of discovery.

It's early days in this work and some fundamental questions remain. TLS imposes a significant incremental load on the DNS as compared to unencrypted UDP, and while client-to-recursive associations are 'sticky' in so far as a TLS connection can be used for many queries, the recursive-to-authoritative association is not so sticky, so the likelihood that a TLS session would be used for just a single query in this context is very high. If the zone is DNSSEC-signed, then the incremental benefit of Authenticated ADoT over and above channel encryption with unauthenticated ADoT looks marginal at best. If the zone is not signed, then the value of talking to the 'right' server is far more important, but to secure this association then, it is necessary for the name server credentials to be secured with DNSSEC in any case. Rinse and repeat. Or just use DNSSEC!

If the TLS overhead is so high then perhaps the server would like to only use Authenticated ADoT with certain clients, in which case the case for mutual authentication is stronger. In any case it must be stressed that even authenticated ADoT is not a substitute for DNSSEC signing. They address different aspects of DNS security and privacy, and they are not mutually substitutable.

However, there is an issue with the IETF that is exposed in this work. The ADD group has been working on service bindings for DNS to signal the supported set of secure transports for a DNS server. The DNS Privacy Working Group is working on this topic. And now there is DANCE, attempting to apply DANE for network client authentication. And there's DNSOP of course. One should never underestimate the potential for the IETF to solve the same problem thirty times in thirty different ways. Just think 'IPv6 Transition Mechanisms' to understand just how well the IETF can explore and standardize every possible option, to the inevitable confusion of all!

## DNS Privacy — DPRIVE

The topic of Authenticated ADoT is a common topic in DNSOP and the DPRIVE working groups, so it makes some sense to now turn our attention to the DRPIVE Working Group at IETF 112.

### RIP DNS over DTLS

There was a time when folk gave presentations on a protocol called DTLS, or datagram TLS. This was a proposal to use TLS over UDP rather than TCP. DTLS had its problems, not the least of which was a comprehensive intolerance of IP fragmentation, so somehow the TLS material had to be crammed into unfragmented UDP packets. While the protocol was written up as an RFC (RFC 7350), and there were optimistic presentations that proposed DNS over DTLS as a means of performing DNS over encrypted sessions without the overheads of TCP, the reality was far more mundane as there were no known working implementations of DNS over DTLS.

It's now proposed to put DNS over DTLS out of its misery and reassign UDP port 853 for use by DNS over QUIC. Finally.

### DoTting the 'a's — TLS for authoritative nameservers

The DNS query/response protocol is used in all DNS contexts. It's used between the end clients and their selected recursive resolvers, and it's used between recursive resolvers and authoritative name servers. No change. So once an encrypted transport profile is defined, such as DNS over TLS, or DNS over QUIC, then it's possible to use this DNS transport in any context, including the context where DNS recursive resolvers are querying authoritative name servers. So why is the DPRIVE working group spending time looking at encrypting DNS transactions between recursive resolvers and authoritative name servers as a distinct area of study? What's left to do?

I guess that the answer is along the lines of 'because you *can* does not necessarily mean you *should*!' Part of the issue here is attempting to defray the overheads of establishing a shared state between the two communicating parties that can be used over a period of time for a number of queries.

In the stub-to-recursive context, the client's stub resolver can set up a secured session that can then be used thereafter. However, this is not necessarily the case for the recursive-to-authoritative scenario. Here, the potential to defray the overheads of setting up a secure session through using the session for the sequence of subsequent queries is far lower in all but the busiest of resolvers and the servers for the most populated of domains. It's also the case that there are no personal details here, as the only end identities in the DNS transaction are the infrastructure identities of the resolver and the server. So, what exactly is being obscured here? Why? And to what benefit? To put it another way, ADoT has an unclear motivational use case. Aside from the benefit of concealing the query name from onlookers, there is little else that is being concealed here.

This is not quite the complete story, as TLS offers both channel encryption and end entity authentication. TLS sessions can be set up in unauthenticated mode where the identities of the end-entities are not authenticated, server authentication mode where the recursive resolver authenticates the identity of the server as part of the session setup, and mutual authentication mode where both the identity of the server and the client are authenticated by each other.

To put it bluntly, the Working Group has not come to a consensus as to what it wants here. And while there is no clear concept of what it wants it does not make much sense to push out specifications of a recursive-to-authoritative secured DNS transaction. There are proposals to simply use unauthenticated ADoT (or UADoT) or use Authenticated DoT, or A2DoT) and no clear idea about which approach the Working Group should work on.

## Unilateral DNS Probing

How could a recursive resolver probe the capabilities of authoritative servers without performing explicit signalling? The concerns here are increased latency, increased resource consumption, and accidental data leakage. It is proposed that authoritative servers listen on TCP port 853 with DoT and UDP port 853 with DoQ, and at this point use any X.509 certificate, ignoring the SNI field in the TLS exchange. It is proposed that recursive resolvers probe for DoT and/or DoQ using the IP address of the authoritative server (and not its DNS name as the appropriately named delegated name server for the zone).

How could explicit signalling improve this? The types of questions include the authentication model, namely X.509 using the WebPKI or a DANE TLSA field with the public key without a certificate at all. And what should happen if authentication fails?

Again, this comes down to a use case and motivation question. Is the goal to push as much of the DNS query traffic behind an encryption envelope as possible, irrespective of whether the ends are authenticated to each other? Or is it the goal for resolver clients to place policy constraints on authoritative servers as to the desired level of privacy for certain queries? Or is this a desire for a recursive resolver to assure itself that it is asking the 'genuine' authoritative server, so that it can trust the authenticity of the response, irrespective of whether the zone has been secured or not.

How to perform this encryption on these queries is the easy bit. Understanding why and when recursive resolvers might want to do this, and when and why the burden of the additional overhead of using a secured encrypted session is warranted for both the recursive resolver and the authoritative server is proving highly elusive in terms of understanding the consensus of the working group. If the majority of authoritative servers are unwilling to absorb the incremental load of supporting encrypted sessions with recursive resolvers, then there is little point in this effort. It's not clear if this work is pushing against an open door or pushing against a solid wall of inertial resistance from the existing DNS infrastructure. Personally, I suspect the latter.

## AAADoT or Another Authenticated ADoT

This is about the resolver client authenticating the name server. This requires the recursive resolver to ask the delegated domain's nameserver for the name of the nameserver of the domain (because the parent's NS record is unsigned by DNSSEC), validate the response (which of course requires the zone to be DNSSEC-signed), then query the _dns service record (SVCB) for the name server name (_dns.nameserver_name) and validate this response as well. Then use the profile in the SVCB record to retrieve the TLS profile and the key for the encrypted SNI field. It can be done. But you need to be incredibly patient to await the resolver's actions to discover this data and validate it. And on the Internet, nobody is patient. This is a proof-of-concept that it can be done, but it's a non-starter for general use.

One way of speeding this up is to place secure information in the parent domain about the name servers of the delegated domain. But even that has issues. What is the speed penalty imposed on A2DoT resolvers in resolving any name, whether the server supports A2DoT or not? Given that resolution of a name is a top-down recursive process, should we care about using A2DoT at level N if the parent zone at level N-1 does not support A2DoT? Is it possible to add new retypes to glue records in today's DNS provisioning world? Or new digest types to the DS records? This is a highly constrained space.

This proposal is a new digest field in a DS record, DSGLUE. Its contents are a delegation-following record that overrides any glue records in the zone. This augments the DS record but has no other impact on the DNS provisioning infrastructure. It provides a secured association of the zone's nameservers and the associated IP addresses, secured by signing the record with the parent's zone key. It's not all plain sailing and this augmentation of the DS records is likely to prove problematic. If this is all about the perception of inability to add new child-originated Resource Records into the parent zone, then let's just say that this door has been bolted shut and pile more information into the DS record. But is it really impossible, or are we just too impatient? We are tinkering with a major piece of operating DNS infrastructure that is incredibly slow and expensive to change on the fly and expectations that this could happen on a time scale faster than, say, IPv6 transition, are incredibly naive!

However, would we be waiting for something that isn't going to happen in any case? I personally suspect that ADoT is mission impossible, and it's time to declare defeat and just move on!

Such a definitive action is usually impossible for the IETF to undertake. The normal course of action is to declare the work 'experimental' and hope that the industry catches on to the subtle distinction that 'experimental' is often a code word for 'not a good idea to use with real customers.' Sometimes that happens, and other times all that happens is that everyone else gets confused!

## DANE authentication for network clients everywhere — DANCE

DANE, or Domain Keys in the DNS, was a fine idea. But while it might be a fine idea, so far it has managed to get nowhere as an alternative to the existing WebPKI infrastructure that supports TLS sessions.

Are there use cases where DANE could be useful? What about cases where both parties to a TLS session want to assure themselves of the bona fides of the other party?

CA certs are a known point of vulnerability in the DNS as much as they are a vulnerability in the web and other areas that use TLS. There are various ways for CAs to be misled and coerced to issue domain name certificates under falsified conditions. We have been unable to clean this up with any success, and the latest efforts with automatically issued domain name certificates based on automated proof-of-control tests has not improved anything. Quite the opposite, it appears.

Doesn't DNSSEC solve this? Well, no. Validation is not implemented at the stub, but when different DNS records get sent to different clients, DNSSEC has its weaknesses — it's not a good authorization tool.

The fundamental premise in many network transactions is that promiscuous servers are exposed to potential DDoS when the service queries are presented over a TLS session. One way to mitigate this is to allow the resource to identify the querier and prioritise the resources at the source to respond to the querier based on local permission policies. Right now, for example, much of this is based on client source IP addresses and in a world of IPv4 NATs and IPv6 privacy addresses, it's simply messy. The assumption here is that there is value in an authority model that will allow a server to allocate resources to an identified client as much as there is value to a client to identify and authenticate the server.

The problem is that web applications typically live behind a TLS offloading front end. The TLS termination is at the cloud front edge while the other functions are performed on the application server platform. A common approach is to use a private root certificate and maintain a separate PKI. DANE offers a solution that sidesteps the private PKI provisioning side effect (anyone who has encountered the combination of Chrome and private PKIs has experienced this problem!) DANE offers a certificate to a domain name binding without a PKI, or even a simple key-to-name binding. The latter approach is a key-only approach where the DNS provides the secure association of name-to-key as distinct from X.509 certificates and a PKI.

Will this effort to provide an efficient form of client authentication be enough to revive DANE's prospects? I'm not holding my breath, but I can see others are more hopeful than me!

## Adaptive DNS Discovery — ADD

The use of DNS-at-a-distance with the inexorable rise of these open DNS resolvers has opened new worlds of potential misdirection and deception. For example, the response to the rise of Google's 8.8.8.8 service in Turkey following a wave of DNS filtering directives was the installation of false routes for the route prefixes used by Google's service to redirect the DNS traffic to local filtering DNS resolvers in any case. The issue in ADD is not only how to discover where your selected DNS resolver might be, but how do you know that you are sending your queries to where you intended?

The DDR work makes use of the SVCB resource record to assist in verifying the IP identity of the remote DNS resolver. The requirement of the resolver's certificate needs to include both the IP address and the resolver's name so that verification of either value type (name or address) can proceed. The SVCB response should provide both the service name, IP hints, and ALPN service profile information. This allows the subsequent TLS connection sufficient information to validate the remote resolver's identity. All this sounds a lot simpler than it actually is, and as expected, the Working Group explored a set of diversions and minutia to get there.

Can DoT be used in a situation where there is a forwarder on the path between the stub resolver and the remote recursive resolver? The basic answer is yes, but only if the stub resolver (or end client) is willing to accept an unverified attestation of identity!

The other part of the Working Group's conversation concerns the scope of applicability of the work. Much of this work assumes the environment of the public DNS, whereas when enterprise and other forms of split-horizon and private name spaces intrude then the exercise is exposed to a further set of complications and caveats. The discussion in the Working Group is about adoption of this work, and whether the working group is amenable to opening the scope of the work to various forms of split horizon and enterprise environments. Part of the issue is why would a private space expose selective parts of the private space to the public infrastructure in order to bootstrap up some verified secure association. The alternative view is that private spaces should remain private and not intrude in any way to the public infrastructure and public name space. I can see a lot of sense behind this ‘keep the fence between public and private intact’ view.

## IETF DNS work

There is a lot of work on the DNS in the IETF these days. The Working Groups are finding it tough to host considered conversations over choices and design of these various aspects of the DNS and its operation, while at the same time accepting all the incoming proposals for work. The current answer is to try and split up the work into distinct Working Groups, but in some cases, it’s become not a case of ‘divide and conquer’ but ’divide and multiply’ instead, where the same work is considered in two or more Working Groups.

However, this does illustrate what I feel is a critical observation in today’s Internet. Addresses, and the routing of address prefixes, is increasingly a marginal activity. The client/server network has been further compressed and fragmented into a client/data centre network, and transport considerations, including unique addressing and routing, are being relegated to little more than historical interest. The glue of today’s Internet is the name space, and the way we negotiate that space. The way that we invest trust in the space is now the core conversation of today’s Internet.

The DNS is taking up so much of the IETF’s collective time and attention because the DNS is the central engineering focus of today’s Internet.

## Disclaimer

## 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*