The ISP Column
A column on various things Internet

March 2024
Geoff Huston

DNS Topics at IETF 119

The Internet has changed quite radically in recent years. The proliferation of service and content delivery platforms at the edge of the network, fed by privately operated feeder networks has reamed out the transit core of the Internet. Most of the traffic and the overwhelming of value within the Internet now transits the short path across an access network from a content platform to the consumer.

Why am I saying this at the start of a report of DNS standards activities at the recent IETF meeting? Because this change in the architecture of the networked environment implies that the essential components of the Internet's common infrastructure are no longer addresses and routing - it's names. The Internet is rapidly shifting to a name-based network and the DNS is now the underlying technology that lies the core of today's network.

So, let's see what we are currently thinking about in terms of names and the DNS at the recent IETF meeting.

The work on the DNS is spread across four working group in the IETF. The DNSOP (DNS Operations) Working Group still handles most of the DNS standards work and appears to be the default place for working on all things DNS. There is also DNSSD (DNS Service Discovery), a working group looking at autonomous DNS-based service discovery in the wide area context. ADD (Adaptive DNS Discovery) looks at the way clients can select a DNS recursive resolver that meets their particular needs or constraints. DPRIVE (DNS Privacy) continues to work on improving the privacy stance of DNS operations. And there was also a BOF on DELEG, looking at ways that could improve the flexibility in the ways a traversal of the DNS distributed database works can be undertaken by recursive resolvers. Here I will just look at the mainstream of DNS work in the DNSOP Working Group and the related BOF session on the DELEG proposal.

DNSOP

Completing Work Items

RFC 9520

The work on caching resolution failure has resulted in the publication of RFC 9520 "Negative Caching of DNS Resolution Failures", which describes a new behaviour for resolvers to cache the information that a DNS server is unresponsive for a particular query over particular protocol. This change to recursive resolver behaviour is intended to moderate the observed storm of queries that often ensues after some form of resolution failure at a nameserver, which has been observed to occur at a level of intensity ten times greater than normal query loads.

RFC 9499

The DNS has been around long enough to develop its own unique terminology. For example, I don't know of any other context in which the rather quaint term *bailiwick* is still being used! (It's a late middle English term that has been steadily dropping out of common use since 1800. The efforts by the DNS community to revive

this term have made absolutely no impact in this somewhat obscure word's general metrics of use, according to Google Book's Ngram Viewer.) RFC 9499 is the latest incarnation of this glossary of DNS terms.

draft-ietf-dnsop-dns-error-reporting
The DNS is deceptively complex, and the introduction of DNSSEC added to this complexity burden. How does a zone administrator made aware that the zone itself has issues that prevent validation? The approach described in draft-ietf-dnsop-dns-error-reporting proposes that a validating resolver can automatically signal an error to the monitoring agent specified by the authoritative server. The error is encoded in a DNS query name, and sending the query reports the error. This specification is in the RFC Editor queue.

draft-ietf-dnsop-avoid-fragmentation
The advice in draft-ietf-dnsop-avoid-fragmentation is now in its 6th year of active consideration. You might think that a simple admonition to keep UDP responses in the DNS below 1,400 bytes and use truncation to trigger re-query over TCP would be a simple piece of advice, but there is evidently more to it than that. The draft has broken this down into 11 recommendations and includes detail on security risks as well as a summary of implementations. I guess this simple advice will be published at some time, but so far, this advice appears to be generally ignored in the deployed DNS infrastructure. I'm cynically confident that the DNS world is as capable of ignoring advice in an RFC as it is in ignoring advice in draft, in industry-wide "flag days" and other forms of advice! The protracted saga of BCP 38 and source address spoofing tends to support such a cynical view.

draft-ietf-dnsop-zoneversion
It is the current fashion in the DNS to pack information into single responses rather than rely in the client making separate queries for each item of data. A case in point is draft-ietf-dnsop-zoneversion, where the use of an EDNS option in a query will cause a server to place its SOA value into the EDNS part of the response. This measure is intended to aid DNS monitoring and debugging by directly associating the data and the zone version being used by an authoritative server instance together in a single response. This document is being reviewed by the IESG.

draft-ietf-dnsop-dnssec-bootstrapping
Once a trust relationship has been established, then the typical behaviour to change one party's keys is for the old, trusted key to sign over the new key. But how is the initial trust created? It is sometimes said in computer science that all challenging problems can be solved simply by adding a further layer of indirection, and the approach advocated in draft-ietf-dnsop-dnssec-bootstrapping certainly appears to borrow from this approach. For the parent to trust the child's key and install a corresponding DS record, the child publishes the proposed DS records in a third-party zone that is DNSSEC-signed. How does the third party validate the keys for this child zone are authentic? On this critical question the draft is less than helpful in providing answers.

Current Work Items

draft-ietf-dnsop-qdcount-is-one
The DNS protocol is quite a venerable protocol, and there are some aspects that are honoured more in common behaviours in various DNS implementations than they are in the massive collection of RFCs (296 according to https://www.statdns.com/rfc/) that collectively define the DNS. One of these is the unsigned 16-bit field in the DNS header that contains a count of the number of queries in the DNS packet. In theory this could be any value from 0 to 65,535 when the OPCODE field is zero (i.e. its a query). In practice, its either 0 or 1. The document draft-ietf-dnsop-qdcount-is-one is attempting to nail this common practice as another DNS RFC. This could be a really short one sentence RFC, or it could ramble across 7 pages. In current IETF fashion, we've opted to ramble.

draft-ietf-dnsop-domain-verification-techniques
The DNS is used for many things beside the simple translation of a DNS name to an IP address. Some services want to know if the party that is requesting a service for a domain name is the real controller of that name, and the most common way of doing this is to ask the requestor to place a token into the domain name to demonstrate that level of control. It seems to have become a bit of a mess!

```
$ dig +short TXT bbc.co.uk
```

```
"google-site-verification=ITX3CwHXxGVfkCmhF4eSwdfo8h2ZGLAZ3zRpYvZi5XA"
"google-site-verification=RaiMXJBIiFvqXHd43kv_ekzmXT2l8ibq5Xy0mulndvU"
"voUGv5zARbEV516E/S8Ugsy9/FOgDGg4n/rpmKZQRROVOj0+2tgzKw3Tk9+Ks6qVbNKU18KTrR5khxTQutDvBg=="
"miro-verification=1a94b0fef7a6d5136a272d5cb425e8dc034e8cfc"
"apple-domain-verification=jFFO0rdS9IrxgWUR"
"msfpkey=3e29l8m08bqxp19k63t73fj5b"
"dropbox-domain-verification=l5djk65wpy3z"
"atlassian-domain-verification=SQsgJ5h/FqwMTXuSG/G4Nd1Gx6uX2keREOsZSa22D5XT46EsEuyaic8Aej4cR4Tr"
"2RLXso9TrRPyhWOEhYggL0U/r1D+g8H7z9RqDBOmcJjSbj88TobGKimtkCrXZNBkDXQDj89lS4mDskNOJyWLdg=="
"MSUEdqCJpCtrI1JuH2-U"
"MS=ms10378910"
"docusign=a10ad7b6-cf7e-472d-8157-23061f5b5116"
"adobe-idp-site-verification=9b850a4a56e3fac19aea1e0ac5db302e5cefab444cd73519dce1c72ccd4db058"
"Huddle"
"docusign=50f10407-e3e4-4f6a-aae4-712d4eb31329"
"docker-verification=aab67462-78f7-4ade-a86b-358645923430"
"J0kgGm0XqA3/6pLD4DHeC5x/dAduzT809P1Iwx/PRCYvVS32rv75RIHKC2aVz47dJxKhPlxGf3h3KXiL6+dyXw=="
"_globalsign-domain-verification=AQ2dURU9RbDOuheuLrx89LSUlA_btgMS6vmFXngBtE"
"v=spf1   a   ip4:212.58.224.0/19   ip4:132.185.0.0/16   ip4:78.136.53.80/28   ip4:78.136.14.192/27
ip4:89.234.10.72/29   ip4:89.234.53.236   ip4:212.111.33.181   ip4:78.137.117.8   ip4:46.37.176.74
ip4:185.184.237.181"   "   ip4:185.119.233.144/30   ip4:185.119.232.158   +include:sf.sis.bbc.co.uk
+include:spf.messagelabs.com ~all"
```

The more this technique is used by various service operators, then the greater the number of TXT records that are stashed at the zone apex. Of course, a querier can't just ask for a particular TXT record. Each queries gets the entire bundle of records!

This draft proposes to pull these bundles of zone apex TXT records apart by using service specific challenge records, creating specialised records such as:

```
_foo-challenge.example.com.  IN   TXT  "3419sdqa32453243d206c4"
```

The proposal contains a useful survey of currently used techniques in Appendix A, which is helpful, but otherwise the proposal is unclear as to what problem it is attempting to solve, and why. It should be remembered these days that the DNS is quite convoluted in terms of roles and responsibilities. The entity that is the notional zone holder (or "owner") may not be the same as the zone administrator who maintains the zone content, who, in turn, may not be the zone publisher (or publishers). When we talk about "validation" which of these parties is providing validation? Should a validation structure make these distinctions in various name administrative roles explicit?

draft-ietf-dnsop-svcb-dane

The DNS is now the content steering protocol for the Internet (see https://www.potaroo.net/ispcol/2023-09/service-routing.html), and the two most useful records to achieve this in the DNS are the name translation record (CNAME), with its ability to transfer control of an individual name out of its original zone to a target zone that is operated by a service hosting provider, and the service binding record (SVCB), with its ability to define the connection parameters of a service without performing additional DNS queries or other capability probes. There was an effort some years ago to take on the X.509 domain name certificate framework and replace it with direct insertion of domain name keys in DNS resource records, relying on DNSSEC to provide authenticity and currency. There are a number of RFCs describing this proposed practice, including RFC 7671 which describes DNAME itself, RFC 7672 which describes DNAME and MX records and RFC 7673 for DNAME and SDRV records. This work is performing the same function for DANE and SVCB records.

The DANE specification contains the advice that a client should follow the CNAME alias chain to its ultimate target host name and use that target name as the base name for a TLSA query, but where no TLSA records exist for that name, it should also query for a TLSA record using the initial domain name. This work suggests that this is not the desired behaviour and only the target name should be used, as in the following example:

```
example.com.            HTTPS 0 xyz.provider.example.
www.example.com.        CNAME xyz.provider.example.
xyz.provider.example.   HTTPS 1 . alpn=h2,h3 ...
xyz.provider.example.   A     192.0.2.1

_443._tcp.xyz.provider.example.   TLSA …
_443._quic.xyz.provider.example.  TLSA …
```

draft-ietf-dnsop-cds-consistency
Whenever the same data is located in two different places in the Internet there is the inevitable question of what should a client do when inconsistency between the various sources of the same data time is detected. This draft proposes that parent-side entities must ensure that updates triggered via CDS/CDNSKEY and CSYNC records are consistent across all of the child's authoritative nameservers, before taking any action based on these records. I suspect that this effort of actively seeking inconsistencies across all possible sources of this data is doomed. How would a single entity detect inconsistences that may occur between the elements of an anycast service constellation? I suspect that the more pragmatic advice, if it is really needed, is to use any instance of DNSSEC-validated data and rely on DNSSEC to provide authenticity and currency and judiciously use TTL records to govern data longevity.

draft-ietf-dnsop-dnssec-automation
In today's DNS world, the function of operating your name's authoritative name servers is generally no longer done in-house. This role is outsourced to a specialist DNS operator. If you believe that continuous availability of your name is vitally important then perhaps a single DNS operator represents a point of vulnerability and multiple DNS operators should produce a more resilient outcome. If the zone in question is DNSSEC-signed then sharing keys across these parties heightens the risk of key compromise, so a preferred option is for each operator to manage their copy of the zone using their own ZSK and KSK keys in a multi-signer configuration.

This draft describes the necessary sequence of steps that need to be performed that combine this multi-signer configuration with the operation of CDS/CDNSKEY automation.

draft-ietf-dnsop-structured-dns-error
This draft has been in the IETF for almost four years, and it still strikes me as a fanciful but totally unrealistic notion! The premise of the document is that a DNS recursive resolver that is applying some form of policy filtering to queries will attach an explanation as to its actions in responding in a negative manner to such queries.

draft-ietf-dnsop-rfc8109bis
This work is about the *priming query* when a recursive resolver starts up with an empty cache. The resolver will query one of the root servers with this initial query to load the resolver with that current set of name servers and their IP addresses that serve the root zone. There are some subtle differences between the handling of this priming query and other queries. For example, the response may not contain all the addresses of all the root zone nameservers, but there is no requirement to set the truncation but to indicate that the response has an incomplete answer section.

This draft is still awaiting some further text, including mention of the DNSSEC-signature that is attached to the name server list in the Answer section, allowing the client to verify the authenticity of the response. The issue of whether the name server zone, root-servers.net should itself be DNSSEC signed has also been raised from time to time.

The issue of DNSSEC signing is "cloudy" in today's DNS. Before DNSSEC the DNS operated the only possible way it could operate, namely that if you sent a query to the "right" IP address then you could believe the response you received. Obviously, this notion has been abused in every possible way over the years. DNSSEC was not intended to prevent such attempts to pervert the resolution process but was directed to allow the client to validate authenticity of the response. It was not material as to how the response was obtained, but that the response was authentic. From this perspective it is not clear exactly what signing the zone of the root zone name servers would achieve beyond what DNSSEC already offers to a client.

draft-ietf-dnsop-rfc7958bis
This document describes how the DNSSEC trust anchor for the root zone of the DNS is published. As DNSSEC clients may use this publication to load its trust anchor, a strict publication format for this information is important. This document aims to carefully specify the means and format by which such trust anchors are published. It's not exactly shattering new material for the DNS but it's an essential part of the overall picture.

draft-ietf-dnsop-generalized-notify
(and **draft-johani-dnsop-delegation-mgmt-via-ddns**)
A critical component of the distributed database that makes up the DNS is the delegation. This is a hierarchical relationship with a parent zone and a child zone. The parent zone denotes this change of authority through the use of NS (name server) records, which also identify the servers that are the authoritative servers for the zone. But the parent is not the entity who determines who are the name servers for a zone, as that is a role undertaken by the child. This means that the child needs to communicate this information to the parent zone on bootstrap and when there are any changes to this information. While many other control functions in the DNS use the DNS itself to manage such operations, this NS information has been handled by various out-of-band methods. In the ICANN-regulated registrant/registrar shared registry model the messages are passed using the Extensible Provisioning Protocol (EPP) (RFC 3730). If EPP is operated over TLS, then it's possible to both protect the information exchange and permit mutual authentication of the two end parties. Problem solved, at least in this cornet of the DNS. But its *clunky*.

DNSSEC permits another method for the child to make available information to the parent that can be validated as authentic and timely. It's also far simpler. Just publish the information in the child zone (and sign it, naturally). For Nameserver records this is already the case, and if the parent periodically polls the child zone for its NS records and validates the response then it could automatically update its own non-authoritative copy of this delegation information automatically. The same approach can work for DNSSEC signature chaining records, where the parent's DS record can be copied from a corresponding signed records in the child zone. This approach, CDS/CDNSKEY is described in RFC 8078. The parent zone operator periodically scans the child zone for CDS records, and if the value of this record set differs from the parent's current DS value, then it can modify parent zone DS record set to match that published by the child. The CDS record is an instruction from the child to the parent to modify the DS record if the CDS and DS records differ. It's simple, robust, and eliminates the need to tinker with EPP to add DS provisioning. If NS records were handled the same way then there is no further need for EPP at all to perform this entire child-to-parent information transfer. If the registry were to perform this scan, then the role of the registrar becomes more of an enrolment operator and less of a relationship manager.

Just one problem. It really doesn't scale very well. We appear to forget that uncoordinated polling (*pull*) as a means of passing information is inefficient. The poller does not know when the information changes, and must poll at a frequency no less frequent than the desired responsiveness of the system. It the model is flipped to a *push* model, where the child notifies the parent of a change in this information. In the DNS we already have not one but two mechanisms to precisely achieve that – UPDATES and NOTIFY. Both have been around for some time and have been deployed in slightly different scenarios. UPDATES are typically used within a zone, it has not been generally used across zone cuts (see draft-johani-dnsop-delegation-mgmt-via-ddns/). The NOTIFY mechanism appears to be better suited to the CDS issue, where the child can NOTIFY the parent's agent of a change in the CDS information, and the agent can perform a pull and install the changes in the parent zone. This approach requires a new RR type, DSYNC, published in the parent zone, to allow the child to determine the target domain name of the NOTIFY message.

This DNS has always attempted to function in a complete manner, where both the movement of data (name resolution) and command and control across the distributed database (UPDATE, IXFR, NOTIFY) are achieved in the DNS itself, eschewing the use of non-DNS control mechanisms, such as EPP.

> In the same was as we sometimes forget that *pull* systems tend to be inefficient and unreliable (RPKI anyone?), out-of-band control systems tend to generate more issues of trust, resilience and timeliness than they solve (the uncanny parallel with the distribution of RPKI credentials again comes to mind!).

All of this skates across the somewhat clunky topic of *delegation* in the distribution DNS database and the role of *authority* in this framework. In the case of nameserver records, which are in effect a *forward pointer* in data terms, the parent's copy of this information is the information used to perform a downward traversal of the delegation hierarchy to resolve a name. Yet the authoritative source of this formation lies in the child zone, as the child needs to retain the ability to manage the name servers that serve this zone's content.

With the introduction of DNSSEC the parent also needed to place a signature of its own over a hash of the child's DNSSEC key. Similar issue, but in this case its necessary for the parent to have information for which it is authoritative to preserve the intended semantics of the signature. This was finessed by having the parent sign over a hash of the child's key, and this hash value is authoritatively published by the parent. But if this is just a hash of data for which the child is authoritative, then howe can the parent claim that it is the authoritative source of this information? Confused? I know I am!

While we are exploring this topic of distributed data structure and pointers the DNS is also missing the reverse of the delegation record, namely the record that informs a client who is the parent zone. A *reverse pointer* in this framework. The assumption in the DNS is that all traversal through the distributed hierarchy is *top down* and the client assembles the set of relationships as performs top-down discovery of zone cuts and delegations. Yet there are a number of places in today's DNS where such a reverse pointer could be helpful.

DNSSEC validation logically starts not at the root but at the endpoint node and traverses back up the delegation hierarchy to the trust anchor at the root of the hierarchy. The issue in the UPDATE and NOTIFY where the child needs to pass information to the parent, has motivated the creation of such a backward pointer in the form of a DSYNC record.

If we were to strip down the question to its basic form, then perhaps we should think about answers to the two basic questions here: "What is the name of my parent and where are its authoritative name servers?"

There is no doubt that the area of the DNS that uses shared data is one of the more operationally troubled areas of in the DNS, prone to misconfiguration and even opens up some windows of exploitable vulnerabilities. In the case of delegation name server records (NS records), the data used by resolvers is the parent's non-authoritative copy of such records in the downward traversal of delegations to find the authoritative name servers for the domain name, while the child's copy of the NS records is the authoritative version of the data. In the case of the delegation signer records (DS records) the authoritative data is published by the parent, but that data is derived from the DNSKEY key value that is authoritative in the child domain. Whenever there is replicated data in distributed system the key question is: What should a client do when the various sources of the data disagree with each other? The DNS has no answers here.

Perhaps the better question is: "How can we ensure the consistency of data that can be presented to a client from different sources within the distributed DNS hierarchy?"

draft-ietf-dnsop-compact-denial-of-existence

There two fundamental performance and efficiency criticisms of DNSSEC. The first is that the inclusion of signed records in DNS responses bloats the DNS response size and this makes DNS over UDP an uncomfortable situation that requires UDP fragmentation or a rapid transition to TCP. The second is that

validation of DNSSEC-signed responses can be very time consuming when using incremental queriers to assemble the validation path.

The DNS response size has been seen as a challenge, particularly in the case of signed NXDOMAIN responses. These DNSSEC-signed negative responses contain the zone's SOA record (to let the querier know how long to cache the negative answer) and its RRSIG signature, an NSEC record that spans the query name, and its RRSIG signature and an NSEC record to prove that there is no wildcard in the zone, and it's RRSIG signature. That's 3 resource records and 3 signature resource records. This is a large DNS response, particularly when the zone is signed with the one of the larger crypto algorithms, such as RSA-2048.

Work by Cloudflare in 2016 pointed out that responding in that manner to a query where the query name does not exist is perhaps more information than what was strictly asked for, and with the multiple signatures there is also the penalty of a larger response. The query contains a query name and a query type. A negative response could indicate that the name itself does not exist, or the query type does not exist. The latter form of response, indicating that there is no resource record of this type for this domain name generates a far smaller DNSSEC-signed response. If the NODATA response indicates that there is no data for this name other than the NSEC and RRSIG signature records, then the client will be unable to distinguish between a no-existent domain and an empty non-terminal name (as the NS records in the parent are not included in the resource record type bitmap for such names in any case). A workaround was to invert the resource record type field in the synthesised NODATA response, indicating that all types other than the queried type may exist for this name. Compact Denial of Existence takes a more direct approach, defining NXNAME, a pseudo–Resource Record type, used in the NSEC type bit map for non-existent names in signed responses. For queries with the DO bit set a query for an empty-non terminal would elicit a signed NODATA response, while a query for a non-existent name would generate a NXNAME response.

This approach to denial of existence, coupled with elliptical curve crypto, can reduce the size of DNSSEC-signed responses to the point that signed responses should fit into UDP and front-end response signers can work even without complete zone knowledge.

However, it assumes that the servers can generate synthetic signed NXNAME responses on the fly, which implies that authoritative servers for a signed domain must have access to the private signing keys for the zone. It also implies that such servers are more vulnerable to random name attacks, as all such random name queries cause the server to perform an on-demand signature computation. Smaller responses on the wire, but at the cost of increasing the computational load on the servers. No gain without pain, as they say.

This NXNAME response has been implemented by Cloudflare and NS1 DNS servers.

draft-hardaker-dnsop-rfc8624-bis

Cryptographic algorithms are a moving target. The "puzzles" posed by encryption are not impossible to solve but are of sufficient complexity that they are infeasible to solve with available computation resources. As computers increase in capability this working definition of "infeasible to solve" keeps on lifting the threshold of what are "safe" algorithms. As this draft notes: "The algorithms identified in this document as MUST or RECOMMENDED to implement are not known to be broken at the current time, and cryptographic research so far leads us to believe that they are likely to remain secure into the foreseeable future. However, this isn't necessarily forever …" DNSSEC leaves the choice of which algorithm to use to sign a zone up to the zone operator. Validating resolvers need to have support for the algorithms used by these signers.

To quote from the draft: "Since the effect of using an unknown DNSKEY algorithm is that the zone is treated as insecure, it is recommended that algorithms downgraded to NOT RECOMMENDED or lower not be used by authoritative nameservers and DNSSEC signers to create new DNSKEY's. This will allow for deprecated algorithms to become used less and less over time. Once an algorithm has reached a sufficiently low level of deployment, it can be marked as MUST NOT, so that recursive resolvers can remove support for validating it. Validating recursive resolvers are encouraged to retain support for all algorithms not marked as MUST NOT."

The challenge here is the concept of measurement of use of any DNSSEC algorithm. The DNS is increasingly opaque to external observers, and measuring the level of support for any particular cryptographic algorithm in the content of the capabilities of DNSSEC validators can be quite challenging (see https://www.potaroo.net/ispcol/2021-11/ecdsa.html for a relatively recent measurement exercise along these lines).

Grease

Many protocols reserve protocol code points that are intended to be used for future extensions, allowing some flexibility in protocol evolution that does not immediately entail throwing it away and restarting with a new protocol for every minor extension to the protocol. Over years such unallocated fields can ossify, and implementations may drop into a mode of assuming that the fields always contain a zero value, for example. In this case when the protocol work returns to extend the protocol the fields that were intended to be used for this purpose are practically unusable. The DNS protocol has a number of such fields, including the Header flags, EDNS Header Flags, Resource Record Type value, Opcodes, EDNS version, EDNS Opt Code and RR Class.

One way to prevent ossification in these fields is to inject random protocol frames which make some use of these unallocated fields, and following up when such use generates an error condition, or "Greasing" the protocol. This proposal is modelled on similar work in TLS (RFC 8701) and QUIC (RFC 9287).

Ranking DNS Data

In a DNSSEC-signed environment all verified DNSSEC-signed data is trustable, irrespective of its source. However, not all data is signed, and in the world of the DNS there are many ways to receive data. The general model described in RFC2181 ranks data from a primary zone file, other than glue data. as the most trustworthy, and then lists lower orders of ranking for non-authoritative data and other additional information provided in responses, including glue records.

The question was asked at the Working Group meeting as to whether it's time to re-evaluate the ranking of DNS data from this RFC, as it is now some 27 years old. I must admit, as a self-confessed "DNSSEC snob I find it hard to understand the point of revisiting this work. If you can't validate a DNS response, then you just can't tell if the data in the response is authentic or not. At that point you are fumbling the dark trying to understand which piece of unauthenticated data or more or less trustable when the risks of a DNS substitution attack apply equally to all such data.

DELEG

The mechanism in the DNS name system that supports delegation is a vital component of the entire DNS architecture. Obviously, any proposed changes to this structure will excite considerable interest in DNS circles, and the DELEG proposal is no exception. I have recently looked at this proposal to redefine delegation in the DNS (https://www.potaroo.net/ispcol/2024-02/deleg.html) and since then the proposal has been aired in a number of DNS-related venues. The question being put to the IETF is to consider whether to create a Working Group to facilitate standards work on this topic.

There are a number of limitations in today's DNS which for some are becoming increasingly irksome. There is no method in DNS referral responses to signal the DNS transport capabilities that are supported by the delegated zone's nameservers, such as DNS over TLS, or DNS over HTTPS. The delegation information provided by the parent zone can only be taken on trust. The authoritative source of the delegated zone's name servers is the child zone, and parent-provided information can only be provided on trust, and in a DNSSEC-unsigned manner. It would be desirable to have a more efficient way to signal the availability of alternative query transports, including secure transport. These days many zones use outsourced operation of the zone, and the tools to support the role of this additional party in the existing registrant. registrar and registry framework are somewhat inadequate. Some folk consider it desirable to authenticate the nameserver that they will be querying, and there is the consideration of future requirements, so some form of extensibility in such mechanisms is also desirable.

There is a visible tension in the DNS over DNSSEC. On the one extreme is the unsigned view, which still uses the maxim that if a DNS client directs the right question to the right IP address, then the answer will be authentic. Which is probably true in a largely benign trustable Internet. But that's not the Internet we have, and in today's toxic world riddled with exploited vulnerabilities "trust" is a highly devalued term. The other extreme view is to place trust in the interlocking chained signatures used by DNSSEC. If the zone you are querying is signed and you can build a chain of trust from the root zone to your response, then it's quite reasonable to conclude that your response is current and authentic. The question DELEG asks is: Does it matter HOW this answer was "discovered?" One view is that the way the answer was discovered does not matter at all, as its the DNS response that you are implicitly trusting when you use the response, and if you truly care about the authenticity of responses from the DNS then you should be using DNSSEC signing as an authoritative server and validating DNS responses if you are a recursive or a stub resolver. Another view is that if the query zone is unsigned, then the next best approach is to attempt to ensure the authenticity of delegations encountered in the name resolution process. I must admit that I find it hard to appreciate this latter view, in that it imposes additional work on resolvers to avoid the additional workload incurred by publishing a signed zone in the first place, but maybe I'm just being a DNSSEC snob in holding that view!

There are some further requirements relating to being able to detect stripping of delegation information by an active middle-attack, preventing attempts to perform forced crypto downgrade and backward compatibility with resolvers that are not aware of a new delegation capability.

In many ways one part of the discussion over DELEG is exploring an alternative to the scenarios considered in the previous DPRIVE discussion about introducing encrypted transport to the recursive-to-authoritative context. The DPRIVE work has come up with RFC 9539, which is an opportunistic approach defined in an experimental specification. The recursive resolver probes an authoritative server using the desired encrypted transport (DoT or DoQ) and if successful caches this result for some three days. An unsuccessful probe result is cached for 1 day. the DELEG proposal includes replacing this opportunistic probing with a clear signal of capability that the authoritative servers support queries over encrypted transport.

It's not only signalling the availably of encrypted transport to query a zone's authoritative name servers. The service industry has enjoyed considerable success in outsourcing content management through the CDN model, and the role of the DNS CNAME alias mechanism has been central to the design of such outsourced services. This alias name form allows a label to be lifted out from its zone and replaced with a new name that can be under the control of the service operator. The DNS operators want a similar level of control. They want to have some autonomy in the management of the name service for a zone, but do not want to step into the shoes of the zone administrator and impersonate them to the parent zone (or registrar) every time they want to change the characteristics of the servers for a zone. There is some expectation that the inclusion of an alias for name servers would provide a similar level of autonomous control for outsourced DNS zone operators.

There is also the prospect of improving the somewhat contorted position with respect to the 'glue' records that are the IP addresses of the name servers for the delegated domain. IN using a service binding (SVCB) record type it is possible to include the IP address(es) of the name server as a *hints* entry in the record.

The DELEG proposal positions the parent as being authoritative for this delegation, so that the contents of this delegation may be signed by the parent zone. If the client is so inclined it can validate the authenticity and currency of this record, thereby resisting the efficacy of substitution attacks where the attacker attempts to substitute false delegation information in a referral response, misdirecting the client, assuming of course that the parent zone if DNSSEC-signed.

The DELEG model with its alias capability comes the ability to remove the levels of implicit representation that exist in the current delegation structure. The name server attributes can be described by the operator of the zone in which the name servers are defined, and not represented indirectly in the zone which is being served by these name servers.

All of this represents a compelling package that can improve the longstanding issues with delegation in the DNS, but it does not come for free.

A new delegation record type that is parent-side only appears to me to be opening up the door to a whole new universe of inconsistency across name delegation boundaries. It does not solve the shared information issue, and one view of weaknesses of the existing delegation model starts with the inability of a child to update the DS and NS records directly.

I get very nervous when folk claim that this work is the start of "DNS v2.0", as this has not been an area where the DNS has had a rich history of success. This is not the first foray into the mechanics of delegation and the associated issues of the zone transfer of authority between parent and child. Earlier IETF work on DBOUND (an effort to replace the clumsy external Prefix Suffix List with a clear in-band DNS signal) failed because everyone wanted DBOUND to do subtly different things, and there was no consensus way forward.

If we take all those motivations for DELEG (clearly signalled encrypted transport, signed delegation information, aliasing for name servers, service hints in place of glue records) and possibly mix in some further concepts, including resurrecting the administrative boundary work of DBOUND, then why do we think that things will turn out differently this time around?

BOFs are always a bit of a worry in my opinion. The entire world of unresolved issues, inconsistencies, differing motivations and perspectives all walk into the room, and clarity and focus often gets lost in the ensuing confusion! While it's tempting to try and grapple with all of the issues all at once, such a response often fails at the first hurdle. Alternatives of more modest incremental adjustment have been a feature of the DNS for the past three decades or so, and while such an effort might lack dramatic flair, it makes up for it in bringing a higher level of assurance that work will be successful in bringing improvements to the DNS infrastructure to match current requirements. To stretch an analogy well beyond its breaking point, should we be ambitious and try to boil the ocean, or just look at heating a rock pool to start with and see where it gets us?

Where is all this heading?

As usual, there is a lot of activity happening in the world of the DNS. Much is this activity is incremental detailed work, proposing small incremental changes to improve some aspects of the behaviour of the DNS and its capabilities.

The work on re-visiting the nature of delegation in the DNS also shows a willingness within some parts of the DNS community to look at some of the more fundamental aspects of the DNS architecture and ask whether the behaviours of the system can be altered. Despite my personal misgivings about the prospects of an IETF Working Group taking on such a broader agenda, I'll suspend disbelief for a second and look at where such questions may lead. Assuming that we really want to break out of incrementalism and start looking at bigger questions for the future of the DNS, then there are a number of conversations about the future of the DNS that may be worth having.

Do we really need to use UDP transport for DNS queries? The web appears to be working perfectly fine using TLS over TCP. Why is the DNS a "special" application that mandates the use of UDP? I suspect that if we were to start today and design a name system for the network in the context of today's intrinsically hostile digital environment it would be very challenging to argue against starting the design process from a transport foundation of TLS over TCP or similar.

If we are using a streaming transport, then why can't we assume a different mode of operation of DNSSEC, where the validation proof is provided along with a signed response? This would eliminate the two most common problems we face with DNSSEC, namely the risks of cramming large responses in UDP and the time taken to build a validation chain using incremental queries. If we dispense with UDP then the entire concept of avoiding large responses is negated. If we can use large responses, then the validation material that allows a client to validate a response using only its local copy of the root zone KSK as input can be bundled with the response. This pre-packaged approach to validation is largely equivalent to the operation of TLS, where the chain of certificates that permit the client to validate signed data using only the local trust anchor as provided input. Why can't the DNSSEC validation proof be stapled to signed DNS responses?

CDNs have completely changed the architecture of content provision. Content is loaded close to the user as a pre-provisioned function, so the user transaction to load the content occurs over a limited distance. Eliminating distance in the content world makes the web service a lot faster and a whole lot cheaper. If the DNS data is signed, then why can't we do the same with DNS data? We don't need a limited collection of secondary servers. We could provision the DNS data into the CDN systems as signed objects, and serve them by name.

If we regard the DNS through the lens of content distribution approaches, then we can use *push* as well as the current model of incremental query-based *pull*? Pre-provisioning clients with DNS data makes giant strides in the speed and privacy of our use of the DNS.

I suspect that we use the DNS in its current mode of behaviour because that's what we've always done, and we've confused habit with optimality. The DNS is clunky, slow, error prone, replete with hidden complexity and unreliable. It behaves in this way because that's the way it's always behaved. If we really want to chat where we would like to go with the DNS, then maybe we need to ask these more challenging questions about alternative ways of operating the network's name infrastructure.

Internet technology is not a collection of immutable RFCs, chiselled into granite tablets. Technology is a conversation, where what we know and what we can do today is balanced against what we would like to do tomorrow and what we might what to understand in order to get there. So far, we appear to have been generally effective in sustaining conversations about the detailed minutia of micro-behaviours in the DNS. Our challenge is to try and talk about some of these larger topics while avoiding fragmentation of such a conversation into a confusing mess. The "design by committee" approach has never enjoyed much success in other realms (need I mention ATM, or perhaps IPv6?) so while these bigger questions appear to have some purpose and potential benefit, maybe we should approach them very cautiously and think as much about how to proceed as to where we'd like to go.

.

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