The ISP Column
A column on things Internet

Geoff Huston
July 2017

Notes from IETF 99 – DNS Activity

Interest in the DNS appears to come in waves. It's quiet for a few years, then there is a furious burst of activity. We appear to be in the middle of a burst of activity, and there is probably enough material presented at the recent IETF meeting to cover the DNS-related meetings in a single article. This interest in the DNS appears to be motivated by three major areas of concern. The first is trying to 'harden' DNS and make it more resilient to various forms of malicious attack, the second is trying to stop DNS from being such a massive source of leakage of user information, and the third is trying to accommodate the needs of the new class of CDNs who are trying the use the DNS as a short cut to link users to "nearby" content sources. The DNS sessions at this IETF meeting appeared to have a little of all three of these areas of interest threaded through the DNS-related working group sessions.

DNS Privacy - DPRIVE

Now that QNAME Minimisation specification has finished, the DPRIVE Working Group's attention has turned to looking at ways to cloak DNS queries in some form of encrypted privacy on the transport channel. We have seen DNS over TLS over TCP, and also DNS over HTTPS (i.e. DNS over HTTP over TLS over TCP). We have also seen reference to DNS over DTLS, which appears to be a UDP-based form of TLS, but the caveats over the inability to perform UDP packet fragmentation in DTLS are an unresolved issue with this approach. Do we have enough candidates to find an acceptable set of options for secure channels for DNS transactions? Or, as was facetiously remarked in one session, should we spin up a new DNS Working Group for DNS Over New Transport, with the helpful acronym of "DONT"!

In the DRPIVE session Christian Huitema presented on his work on DNS over QUIC. QUIC is a TCP/TLS 1.3 amalgam introduced to the IETF by Google over a year ago. QUIC looks in the outside like a UDP protocol with an encrypted payload, while on the inside it's a TCP session with some of the latest TCP flow control and recovery techniques. QUIC is NAT agile, it allows parallel streams, and 0-RTT resumption of a stream. When used for DNS queries and responses QUIC can permit a separate stream for each query, retransmit efficiently, permit arbitrarily long messages, and all this with no head of line blocking. QUIC encrypts its payload and can authenticate the remote party. All this makes QUIC remarkably agile and it looks like an ideal fit for DNS. QUIC appear to offer the DNS a service which is at least on a par with UDP in terms of functionality and efficiency when looking at stub to recursive resolver transport, and certainly in many contents it looks like a better proposition in terms of being able to offer encryption and handle arbitrarily long responses without packet fragmentation, truncation or fallback to TCP. One thing Christian points out is that QUIC is User Space not kernel space, so it's more agile for protocol change. Also, there is a conversation about how could QUIC be improved in areas such as multiple encryption frames per UDP packet in QUIC. Part of the motivations of Christian's work is whether there are considerations in transporting DNS that could assist in the development of QUIC as much as the DNS itself. All of this seems to point to a Too Good To be True proposition for DNS over QUIC, at least out there on the edge, looking at how DNS clients can talk efficiently and in secret with their chosen DNS recursive resolver.

Within the discussion of this work, the point was raised as to whether "we are designing a new crypto transport protocol for the DNS" or "we in fact designing a new DNS protocol itself". The one thing that is teased out is that this privacy topic brings out a practical consideration of differences between stub (client) to recursive resolver and recursive resolver to authoritative name server transactions, and it appears that their practical operational requirements, particularly as they relate to channel encryption do indeed differ.

QUIC is an exercise in hiding not only the payload, but also the flow control communication behind the encryption. The question then should any part of the DNS transaction be carried in the clear? For example, Google operate a DNS over JSON over HTTPS (https://dns.google.com), and in this case the port used for the query and the response is port 443. To an external observer who is not privy to the session encryption keys, a session with this server would look much like any other HTTPS transaction. Only the server's IP address used to access the service might indicate that this is in fact a DNS transaction. If you also make use of EDNS padding options to mask the payload lengths then this further exacerbates the task of detecting the fingerprint of DNS traffic, even when encrypted.

DNS over TLS proposes using port 853, making the transaction clearly evident, even if the content is opaque. If we really want to bury the DNS then perhaps we should consider some further measures. One is to multiplex both DNS and HTTPS transactions on port 443 on the same server. But if you are not using conventional TCP port addresses as the demuxing signal, then how can a server tell the difference between DNS over TLS on 443 and HTTP over TLS on 443 and correctly de-mux the two traffic streams? It has been observed that the text of the initial frames differ in the first 14 octets - so demuxing is indeed trivial. The issue is that there is a real concern about overt metadata and the reaction is that it is possible to code in a more covert manner. The larger issue is whether we (for some large value of "we") are willing to go forward with this? If network operators and eavesdroppers want to deploy middleware to observer or even manipulate network transactions, then application designers are motivated to want to hide these transactions.

Now the conversation becomes one that looks at how much metadata is placed in the clear, and whether the eavesdropped and middleware-intercepted Internet has become so toxic that all that is left of the end-to-end network is a narrow aperture of port 443. Consequently, we now appear to be looking at how we can perform demuxing of various services from signalling within the encryption envelope. Perhaps we have managed to limit our transport choices to QUIC or TLS over port 443 as the universal substrate irrespective of what service we may be trying to access.

DNS Operations

Since the closing of the DNS Extensions working group much of the work in devising potential extensions to the DNS has migrated to the DNS Operations working group. Frankly I'm not sure why the DNSEXT working group was closed in the first place – as is evident, the work proceeds in any case and the choice of which working group is perhaps an opportunistic choice as to who is prepared to accept it! So right now much of the work in devising new EDNS options and new RR types and similar is using DNSOPS as the working group of choice. The mantra of "whatever the problem, the answer is to just stick in the DNS!" is subtly changing to: "We can a EDNS option for every problem!" Which means that the DNSOPS agenda is packed with a bunch of interesting ideas, some potentially useful and some just toxically bad! Here are a few that piqued my interest.

A longstanding limitation of the CNAME alias function in the DNS is that it was not possible to place a CNAME record at the root of a zone. Enter the ANAME RR, which explicitly permits an alias to be placed at the root of a zone. The ANAME also differs from the CNAME in that the server is permitted to provide the response instead of just the referral. This appears to be an instance of the proposed set of tweaks to the DNS to respond to the needs of CDNs. There are some subtle impacts of the server-side de-aliasing of the name, including the distinction between zone owners and zone

publishers and the consideration of DNSSEC, in so far as the alias result is signed within the validation path of the alias name, not the original name.

Work continues on the concept of a DNS "session". If we are going to use DNS over TLS or any other form of streaming protocol, then it is highly inefficient to bring up a new TLS over TCP session for each query, particularly in the context of the stub to resolver relationship. Like many aspect of the DNS backward compatibility rears its head here, as there is already a set of diverse resolver behaviours relating to management of TCP sessions across Bind, Unbound, Knot, OpenDNS, and Google. The major question at the moment is whether this signalling should be performed in a way that is consistent with all things DNS sp far, by using a Resource Record type, or make a clean break and encode session signalling in a TLV option in the control stream.

The reverse DNS space (translating IP addresses to names) is often tedious to populate in IPv4, and if that's tedious, then IPv6's reverse space is hopelessly so. Couldn't we pass a server a number pattern and get the server to complete the response from a template? (Assuming of course that you can perform on-the-fly DNSSEC signing.) BIND has had $GENERATE for many years, and this work is an effort to simplify and generalise the approach to include the functionality in the standard DNS server repertoire.

One of the observations about attacks on DNS servers is that the DNS name does not go "black" immediately. Recursive resolvers continue to serve from their caches until their cache timers expire, and at that point in time they will remove the object from their cache if the authoritative name servers cannot be contacted. Under attack the names "goes dim" as more and more recursive have their local caches expire over time. A precise reading of the specification reveals that the TTL field in a DNS zone defines the maximum interval before refresh of a cached copy of a name, but that is not quite the same as the period to retain an entry before cache deletion. What if resolvers kept stale data in their cache for a longer period if the authoritative name servers were non-responsive. The implication is that the recursive servers would be able to keep a name 'alive' through most current DNS attacks, and an attack would need to last for a period this stale data time plus the TTL if the attack were to successfully completely black out a name. IPR issues? Yes, from both Akamai and Google.

A related approach is "opportunistic refresh". This is based on the premise that every update in a DNS zone is accompanied by an incremental change to the zone SOA value. This means that if the SOA has not changed, then the cached entries are still valid, and can be assumed to continue to be valid for a further refresh interval based on the time of the latest SOA fetch. Again, using the Texas Chainsaw Massacre of the DNS, EDNS options, the client can add a EDNS option to the query to request that the response includes the SOA RR in the additional section, and the server adds both the SOA record and the EDNS option in its response. The response of the EDNS option is to signal back to the client that it will be accompanying every zone change with a SOA update, so this opportunistic piggybacking of zone change information can be applied to all entries in the local cache that relate to this zone.

The DNS is a simple protocol, and its used in an even simpler manner. The client asks a single query, and it gets an answer. The server will often add additional information, particularly in the case of top-down search where the response will often list not only the next name servers to query, but their IP addresses as well. But this additional information is not directly trustable, and credulous DNS resolvers that loaded all information passed this way into their cache could be easily misled. So the first exercise is to understand the merits of getting recursive resolvers to apply DNSSEC validation to this additional information, which, in turn, requires the server to facilitate this by passing back the digital signatures of this information in addition to the digital signatures of the answer. One proposal is to extend this by equipping the server to provide additional responses in the same DNS response envelope. For example, a name server may choose to add a DANE TLSA record to the response for a query for an A or AAAA record of a name. In this multiple response model the server is adding information to the response that it believes will be helpful to the client. A related proposal is for the client to pack multiple query types into the same DNS envelope. For example, a client may want query for both the IPv4 and IPv6 address records of a name, and using this multiple query-type model it could do so within a single

DNS transactions. However, they are not addressing the same underlying issues, and both have a role in today's DNS interactions. These are both optimisations that improve and not damage the DNS interaction, as long as you assume that the Internet is capable of reliably handling larger DNS transactions, and that these larger responses will not be exploited as DDOS attacks. These are of course assumptions that should be questioned!

The DNS masks a lot of information, which some regard as a feature, not a bug. When a query is passed through a recursive resolver, the identity of the original client is lost. There is no "trace" in the DNS and only the recursive resolvers and forwarders are able to create the inferences that link queries to the agent that originated the query. But Content Data Networks wanted to use the DNS for location. They wanted to provide an answer to a DNS query that reflected the "closest" instance of a service. This can be made to work in many cases, particularly when an ISP coralls all customer DNS queries through an ISP resolver. But the model breaks down with the open resolvers, such as Google's Public DNS service, or OpenDNS. This motivated the adoption of the EDNS Client Subnet option, where the network prefix of the original client was carried through these recursive resolvers with the query. That way the CDN could provide what it felt was an optimal answer. But while some parts of the IETF are sensitive to leakage of personal information, this is not a universal theme, and the "xpf" proposal advocates carrying the entire transport 'signature' of source and destination IP addresses and port numbers of the original query as an attribute of the forwarded DNS query. There is the broader issue about the inclusion of metadata into queries and responses and the use of 'trails" in the DNS that work in the exact opposite to the work on DNS privacy - the tension here is quite overt and it's not clear if the benefit of more selective DNS responses outweighs the downside of potential leakage of far more information about individual user browsing habits. Such information insertion could readily facilitate external observation and monitoring. One comment in the discussion on this subject was relating to the Client Subnet option: "If we were thinking of this today, we probably would not have done it." On the other hand, we hear from Some of the larger CDN operators that "we are going to do this anyway, irrespective of the IETF's qualms!" As we have painfully seen with NATs, the IETF walking away from standardising a technology does not stop the development and deployment of the technology – it just ensures that the problem space because larger as there are no standards to assist implementers and operators to all work within the confines of functioning interoperable technology.

It has long been a source of major frustration that the error codes in the DNS are just inadequate to match the range of queries being performed. For example. if a DNSSEC-validating recursive resolver cannot validate a DNS response it returns a SERVFAIL error code, which is essentially a signal for the client to try a different resolver! This proposal is to introduce an EDNS option to add a 16-bit error code that could enumerate all these specific efforts conditions. For example, a validation failure is somewhat different from a trust anchor failure in DNSSEC validation, and a lame delegation signal for the REFUSED response code, and so on. The motivation here is that this gives a client more grounds to decide whether to abandon the query, or not. One of the open issues is whether this EDNS error response would be permitted in a query that did not have EDNS options added in the first place. There is also the issue that they are not protected responses.

DNSSEC is also being considered in terms of changes to the way the DNS operates. There is a push to move on from "simple" public key cryptography based on prime numbers to elliptical curves and more experimental cryptographic algorithms that appear to offer some reliance against quantum computing techniques. Today we are looking at algorithms including Ed25518, Ed448, NSEC5, and PQC. However, performing an algorithm upgrade to a DNSSEC-signed zone can be complicated, particularly when it's not clear that all clients actually support all crypto algorithms. What if the client could inform the server as to what algorithms it supports, and receive responses that include only those RRSIG records that use those supported algorithms. Obviously, this calls for yet another EDNS option in the query and selectively includes signature records that are signed using the listed algorithms.

At some point the DNS ceases to be a simple protocol, but morphs into a programming language in its own right, where the client is specifying particular actions to be performed by a server! I'm sure that someone is already looking at how to use the DNS alone as a way of programming some of the classic

AI benchmark tests. The only consideration now is whether they will come back to DNSOPS and want to standardise their solution!

DNS Service Discovery

There is yet another world of the DNS where instead of using the DNS to map DNS names to some associated attribute (such as an IP address), the DNS is used as a service discovery protocol. Apple's Bonjour protocol is a good example, where multicast on a local LAN is used by a client to send a DNS request for a service (such as a printer) and all services matching the quest would answer with their details. The subtle change is that of a change from unicast to multicast, so that the client's request for services is sent across a local network realm to all local service agents. IPv6 had a similar idea with the Site Local Auto Address Configuration (SLAAC) approach, where instead of the broadcast ARP method, SLAAC uses multicast to find local gateways and configure local IPv6 hosts.

Seems that a lot is riding on multicast. So it was a bit of a jolt for me to see a presentation at the DNSSD Working Group ask the question "why all this non-multicast discovery work?" It noted that on many modern network technologies multicast is expensive and/or unreliable. It noted as examples Wi-Fi, advanced multi-homed edge networks, meshed networks, enterprise networks with large numbers of clients. I could add cellular mobile networks to that list, and doubtless there are others.

What can we do? It seems that we are not prepared to completely abandon multicast just yet, but to make things work we are now heading in the direction of proxy agents and helpers, intended to simulate the multicast functions across a non-multicast environments. I can't help but wonder if we were to design the IPv6 protocol today would we still craft in such a major reliance on multicast as the configuration bootstrap technology?

Author

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

www.potaroo.net

Disclaimer

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