October 2020
Geoff Huston

DNS Trends

We used to think of computer networks as being constructed using two fundamental common infrastructure components: names and addresses. Every connected device had a stable protocol address to allow all other devices to initiate a communication transaction with this device by addressing a data packet too this protocol address. And every device was also associated with a name, allowing human users and human use applications to use a more convenient alias for theses protocol addresses. By mapping names to protocol addresses the realm of human use could navigate the network's services by using symbolic names, while at the level of packets the data flow was directed by the network based on topology information of where these device addresses were located.

But that's 1980s thinking and 1980s network architectures.

Architectures have evolved, and today's Internet architecture has, surprisingly, dispensed with that view of the role of addresses. These days, due in no small part to the exhaustion of the IPv4 address pool, but equally due to an architectural evolution that had to cope with the massive explosion of numbers of devices in networks, we've shifted to a client/server network model where clients initiate connections and servers respond. This means that clients don't require a permanently assigned address. Instead, they can make use of an address only while it is communicating and pass it back to a shared address pool otherwise. Equally, on the server side we've seen the aggregation of uniquely named service points into service delivery platforms, and the multiplexing function that directs clients into the appropriate service rendezvous point is a function performed at an application level rather than as an infrastructure function. We're now using the address infrastructure in very different ways than the way we had envisaged in the 1980's. The Internet's name infrastructure is subject to the same evolutionary pressures, and its these pressures I'd like to look at here. How is the DNS is responding? There are three parts this survey: trust, privacy and all the other stuff!

1. Trust

Can you believe what the DNS tells you?

The answer is that you probably can't!!

This obvious failure in the trust model has been exploited in many ways by many parties. The DNS is seen as an overt control channel. For example, access to a named service can be blocked if that name does not resolve in the DNS. As a consequence, we've seen the rise of deliberate lies in the DNS where content and services that are categorised as harmful are removed from access by withholding the associated name resolution in the DNS. A number of open resolvers have turned this filtering of the DNS into a positive attribute, and there are a number of so-called "clean feed" resolvers that do not resolve a collection of service names where the service is deemed to be harmful or criminal in some manner. Not only is this selective filtering of the DNS a distinguishing feature in the realm of competitive open resolvers we've also seen a number of national regimes placing the onus on ISPs to block certain services, and given that addresses are no longer uniquely associated with individual services the

implementation of these national regulations is invariably performed through DNS blocking. There have also been exercises to attempt to monetise the DNS, where "no such domain" NXDOMAIN DNS responses are rewritten to send the user to a sponsoring search site through response rewriting. DNS lies have also been used in the IPv6 transition environment where the DNS records, the protocol addresses, are synthesised to allow the user to be steered through a IPv4-IPv6 transitional environment. The motives of all these exercises may vary, but the result is the same, in so far as the DNS answer is a lie. Then there are the hostile efforts to replace a genuine response with a lie in order to mislead the user. There is the technique of response guessing to try and insert a fake response before the 'normal' response, as in UDP the first response is this response that is used. There are manipulated glue records and even attacked on fragmented packets. The insidious nature of these forms of attack is that they rely on the host system running quite normally. It's the infrastructure of the name system itself that is being perverted here and the applications are completely unaware of this manipulation.

The response to this need to detect that some form of manipulation has taken place, and, even better, to withhold presenting the lie from the user, is to add a trust mechanism to the DNS to add a digital signature to DNS responses. The idea is that a digital signature attached to a DNS response can allow the receiver of the response to be assured that the DNS information is current, that it is authentic, that it has not been manipulated or altered and it cannot be repudiated. DNSSEC, the framework for adding digital signatures into the DNS was some 10 years in the making. A key pair (or pairs) can be associated with a delegated zone in the DNS, and a set of five further DNS Resource Records (RRs) are defined in this framework to sign each entry and to aid in validation of the signature.

The commentary on DNSSEC deployment varies considerably. Some 25% of the world's users cannot reach a DNS named service if the DNSSEC signature cannot be validated. That's triple the level from early 2014, so there is certainly some momentum in the adoption of validation (https://stats.labs.apnic.net/dnssec). But at the same time, the number of DNSSEC-signed zones appears to be pitifully low. Of the hundreds of millions (perhaps billions these days) of delegated zones in the DNS we see some 8M signed zones in one such survey (https://www.secspider.net). Perhaps a more relevant metric is the ranking of domain names by usage and the ratio of DNSSEC-signed zones in that set. The Alexa top 25 list is a good place to start. None of these are DNSSEC-signed names. A late 2016 study of the Alexa top 1M names found that only 1% of these names are DNSSEC-signed. A similar study of all of all .com, .org, and .net second- level domains found that between 0.75% and 1.0% of all domains in these three zones publish a DNSKEY RR. It appears that turning on validation of DNS queries in a recursive resolver has very little downside given that so few of the popular DNS names appear to be DNSSEC-signed in the first place!

Why is DNSSEC zone signing so uncommon? Are the content and service providers that are behind these DNS names unconcerned about potential misdirection of users? Of course not! They are extremely concerned as ultimately this is all about their potential revenue. Is this ignorance about DNSSEC? Again, not at all! This is a choice. These providers want to prevent users being misdirected, but equally they want to reduce the dependence on intermediaries and they want the user service experience to be as efficient as possible. If DNSSEC was all that was available, then content and service providers would be using it. But it's not the only show in town. These days service provision uses TLS. Almost every service URL out there is an HTTPS URL. Can users be misdirected with TLS? Not normally. Misdirection or deception requires the service provider's private key to leak or it requires the Web PKI certificate system to be corrupted. TLS is also fast, because all the credentials needed to validate the certificate are provided in the TLS handshake.

Is DNSSEC validation as fast? Well no. DNSSEC validation is a serial query sequence all the way back up the name delegation path. Is DNSSEC zone signing simple to deploy? No. There is local key management, the ZSK/KSK key split, limited automation, limited support for high resilience hosting. And the fundamental criticism is that all this additional effort doesn't stop recursive resolvers passing back lies in the DNS for DNSSEC-signed zones anyway!

What? The entire point was that lies in the DNS were just not possible with DNSSEC wasn't it? Well yes, but that's not the way we've deployed it so far. Recursive resolvers perform DNSSEC validation. Stub resolvers that sit over there with the user don't perform DNSSEC validation. Instead, the recursive resolver withholds the response if the DNSSEC validation fails. But what if the recursive resolver is the one that is telling the lie in the first place? The stub resolver is none the wiser given that it's not validating, So the lie stands. If the ISP is blocking some names, performing NXDOMAIN substitution or redirecting actual names, then the stub resolver is just caught in the lie. All that effort to sign the zone, and all that effort in validating the DNS response, yet absolutely no robust protection against being misdirected? TLS just seems to offer a solution that is faster, simpler and more robust. No wonder nobody uses DNSSEC in the service world. It's just a case of more pain, and no real gain.

Is DNSSEC good for anything else? As long as 75% of users sit behind non–validating DNS resolver systems and an estimated 99.9% of users don't directly validate DNS responses in any case, then we cannot place critical information in the DNS in a secure fashion and expect everyone to be protected by DNSSEC. This means that the incentives for putting critical information into the DNS and protecting it with DNSSEC are not looking very convincing. There is simply no natural market-based incentive for deployment of DNSSEC. This is a distressing conclusion, as it would certainly be more useful for the network and its captive user population if its name system was trustworthy.

It has been said a number of times that the heart of many issues with the DNS lies in the choice of a transport protocol for the DNS. The use of UDP as the primary first choice protocol and the fallback to TCP means that its challenging to place large quantities of information in DNS answers while still operating within what we've become accustomed to in terms of parameters of speed and robustness. Validation is a very inefficient process, and the inefficiency is increased by the DNS model where the onus is placed on the client who is requesting the information and not the server who is the source of this information. End clients do not validate because every validation operation would entail further DNS queries in order to construct the validation chain, and the incremental time penalties would be unacceptable in terms of user expectation.

Frustratingly, we know how to make DNSSEC validation faster, and the approach is to pre-provision the validation answers. We can package up all the answers to the DNSSEC validation chain construction queries and include then as additional information to the original signed answer (RFC7901). However, it's unlikely that this is viable in a DNS-over-UDP framework. If we want to go down a TLS-like path and package up a validation chain into the DNSSEC-signed response, then we are inevitably looking at DNS over TCP (or DNS over TLS). The price of this trust solution is significant and that creates a higher threshold for the benefits that trusted answers in the DNS can provide. If all this is about protecting users from a Kaminsky-styled attack, then that's just not enough of a case. The benefit needs to be far more than that to help justify the considerably higher costs in moving the DNS from UDP to a TCP-styled platform.

2. Privacy

Everybody looks at the DNS. Everybody. Because the Internet is funded by its users, then what users do on the Internet is of paramount interest to folk who sell services to users. Because all crime these days is cybercrime then the criminal and abusive behaviour on the Internet is of fundamental interest to those agencies whose role is to police such behaviours. Because the Internet is these days largely about how individuals choose to live their lives and how and why they interest with others then we've learned that what users do is of paramount interest to government. How can you find out what users do? Easy. Look at the DNS. Every transaction on the net starts with DNS query, and the DNS exposes every action. But it's worse than that. The DNS is needlessly amd senselessly chatty. The DNS over-exposes information.

At all points in the DNS these queries and responses are collected, packaged, analysed, profiled, replayed and traded.

How can we make the DNS not the go to system to expose users and user behaviours to business and government alike? How can we improve its privacy?

There was little in the way of motivation to do anything about this question for years. After all, if the Internet actors are busy constructing a global economy based on surveillance capitalism, so why should the parties conducting this surveillance make the task any harder than necessary? The watershed moment that changed the stance for many was the moment of the publication of material gathered by Edward Snowden. Government agencies had spent considerable sums in weaponizing the Internet and transforming it into a highly effective surveillance tool that operated at a scale of national populations. Their motivations were not overly concerned about your future purchases, but more about your personal profile. And of all the components of the Internet, the system that laid out all this information in a clear text pre-packaged format was the DNS.

In response, we've been changing aspects of the behaviour of the DNS to try and stop the most blatant forms of information leakage.

The first of this set of privacy-enhancing responses is so-called Query Name Minimisation. The change is to prevent the DNS name resolution process from being an unconstrained extraneous information leak. This largely relates to the interaction between recursive resolvers and the authoritative name servers. The task of the recursive resolver is to find the right name server to ask, and it starts at the root and asks the query. The root zone server's response will direct the queries to the name servers of the relevant delegated tope level domain name, and this process repeats as the resolver traverses down the delegation hierarchy until the resolver has an answer. But in this process every name server in this sequence, from a root server down, is now aware of the full DNS name that is being resolved. Query Name Minimisation trims the name in these queries so that only the next label is exposed to each name server. Root servers will only see top level domain name queries, Top Level Domain name servers will only see second level name queries and so on.

There has been some further work to understand the most robust query type for this discovery process. The initial suggestion of NS queries has been supplanted by A queries in the light of experience with this approach. The issue of CNAME rewriting and the equally vexed question of Empty Non-Terminal domains and the variable behaviour of name servers in such situations have added some complications to this question. There now is widespread use of this technique of this approach, although some implementations have taken some license with the specification and used their own re-interpretation of the technique. Some resolvers, apparently including Google's public resolver service, only performs Query Name minimisation to the first three levels of the DNS name. It appears as if the recursive resolver is deliberately withholding full query name information from the root servers, and the top level and second level domain name services but is quite willing to disclose the full query name information to servers for zones that are deeper in the name hierarchy. Is this approach motivated by protecting user interests or motivated by an effort of denying information to authoritative servers located at the upper levels of the name hierarchy?

Of course, if we were serious about user privacy, then the Client Subnet extension would never have been specified. The knowledge of full query names that are emitted by a recursive resolver is to some extent mitigated by the inability to conclusively associate such queries with an end user. But if the query is also loaded with the IP address of the end client, or even the network subnet of the end client, then all pretence of privacy protection has been shredded! While Query name minimisation could be seen as a positive step in providing greater level of concealing extraneous information in the DNS, the use of Client Subnet in queries is a gigantic leap backward!

A generic response to privacy considerations in the Internet has been channel encryption. Telnet was replaced by ssh because of the issues of running sessions over the Internet in the clear. Similarly, HTTP has been largely replaced by HTTPS for much the same reason. The DNS is increasingly an anachronism in still passing queries and responses in the clear. Not only does it permit eavesdropping, but it also enables efforts to manipulate the responses, all to the detriment of the user.

However, to repeat an earlier observation, the heart of many issues with the DNS lies in the DNS' choice of transport protocol. Encryption normally involves a number of steps, including the presentation and validation of credentials to confirm that the client is talking to the party they intended to talk to, and also to establish a session encryption key to allow the subsequent data exchange to be encrypted in a manner known to the two parties, but unknown to all others. This is challenging in UDP. The efforts to implement TLS over UDP, namely Datagram TLS (DTLS, RFC 6347)) has the overhead of the exchange of credentials and session cipher establishment, so it's a long step away from a single packet exchange of query and response. DTLS also should avoid IP level fragmentation, but it cannot avoid large payloads associated with this session establishment process. The result is that fragmentation is pushed up to the application layer and DTLS needs to handle payloads that extend across multiple DTLS datagrams. It appears that the additional overheads of DTLS roughly equate to the overheads of TLS over TCP, but with some added fragility relating to packet fragmentation that is not replicated in TCP. The result of this fragility of DTLS means that when we refer to DNS over TLS, we are in fact referring to DNS over TLS over TCP (DoT). It is this TCP-based implementation of TLS that has been implemented and deployed over the path between the stub resolver and the recursive resolver.

DoT adds encryption to the stub to resolver path and encryption not only hides the query and response stream from eavesdroppers but also DoT prevents alteration or manipulation of the response by third parties. The recursive resolver can still lie about the response, and unless the stub resolver is performing DNSSEC validation (and it's likely not!) and the domain name is signed (which it most likely is not!), then any DNS lie from the recursive resolver will be unnoticed, whether or not the transport channel from the recursive resolver to the stub resolver uses TLS or not. A lie is still a lie no matter how secure the case used to carry the lie. DoT it does not eliminate the potential for manipulation of DNS information, but limits the number of entities who are in a position to perform such manipulation and where and how the manipulation can be performed. It could be argued that with DoT all you really know is who is lying to you!

How far should channel cloaking go? Should the identity of the other party be obscured? Should the fact that these are DNS transactions be obscured? DoT makes no effort to cloak its use. The use of TCP port 853 for DoT is a visible signal that there is an active DoT connection. The use of a novel port number is likely to cause many firewall configurations to trigger their drop filters. The IP address of the remote end is clearly visible, as is the TCP header. The TLS handshake may get around to using ECH and encrypt the server name at some point in the future, but in the case of DoT it probably is a minor artefact given that name based overloading of service IP addresses is not happening in DoT today and unlikely to do so in the future.

Is DoT going anywhere? It's unlikely in my view. Right now, it requires users to play with their DNS settings, and that is a massive barrier to widespread use. Some users may use it as a means to jump across one set of recursive resolver's DNS filters, such as those provided by their ISP, to hook up with another DNS provider, but with the overt signalling that this is happening it means that an ISP can readily block this if it so feels. In theory the use of TCP permits larger DNS payloads, and there is a glimmer of hope that we could use DNS chaining to make DNSSEC validation fast and efficient on end systems using DoT. But so many other preconditions, including server provisioning of DNS Chained responses and a reliable way for the DNS to manage large responses, mean that it is still a distant glimmer of a possibility and nothing more.

DoT is seen as a replacement for the existing DNS infrastructure service, where the DNS is a service located on the common platform and applications use the same DNS resolution calls to the platform as they have always used. It's a platform approach to securing the DNS. Adoption is probably going to require some form of automated provisioning that typically involves the local access service provider. Given that the major compromise threat actor here is the same access ISP and given that the ISP is operating the recursive resolver in any case, then it's very challenging to understand the incremental benefit of DoT deployment to an ISP. Perhaps it may be that its benefit is as a barrier to other hosts in the local network. Local residential and enterprise environments are cluttered with IP stacks from many providers. A compromised stack is inside the external firewalls and is trusted merely by its physical location. DoT shifts a DNS host's conversation into a protected channel where the protection is against other hosts on your local network!

DNS over HTTPS (DoH) uses the same TCP and TLS foundations, but adds an HTTP context to the transactions. There are a couple of changes here that are interesting. The first is the switch to TCP port 443. It looks like any other HTTPS traffic and is not so readily identified in the network. Second, the DoH servers do not need to use dedicated IP addresses. Like the web itself, the HTTP protocol allows for named service points. And with TLS 1.3 with ECH even the server name can be concealed in an encrypted envelope. But there is a little more this this. HTTPS is an application level protocol, and this approach allows an application to bypass the DNS services provided by the platform. This means that no ISP-based platform-level configuration is necessarily relevant, and the application can not only conceal its DNS transactions from the local and remote network, it can also hide these same transactions from the platform and other applications. If this heads to HTTPS /3 using QUIC then a number of capabilities are unlocked. Not only is the transport control protocol cloaked behind an encryption envelope in QUIC, a number of DNS requests can be made on a single transport channel simultaneously.

How far can we go with this effort to advance a privacy agenda in the DNS? Once we've deployed QName minimisation, discarded Explicit Client Subnet, and adopted DoH over HTTPS/3 as the application-to-recursive resolver protocol and pushed DNSSEC validation to the application via chained DNSSEC responses, then much of the achievable trust and privacy agenda has been realized. At that point, much of the ability for an onlooker to associate an end-entity identity with a DNS request is severely curtailed, and while a recursive resolver is still privy to these user transactions, the use of DNSSEC all the way to the edge makes response manipulation by the resolver particularly challenging. Should we encrypt the paths between a recursive resolver and authoritative servers? Assuming that ECS has been abolished, there is little that such transactions directly reveal about the identity of the end user, and the larger the pool of clients served by a recursive resolver the larger the crowd each individual's queries can hide in.

3. Other Topics

There are a number of other aspects to the technology evolution of the DNS, and I'll put them all into this section!

The DNS has traditionally used a 7-bit ASCII code for names. Upper and lower case are equivalent, and in addition to the Latin characters DNS labels can use hyphens and number characters. In certain circumstances the underscore is also permitted. The expansion of the DNS into a larger character repertoire has not been a stellar success. The design decision was to preserve the capabilities of the DNS system and use encoding to map the larger character set into this restricted alphabet. The choice of Unicode as the underlying character repertoire was not all that good a choice. Unicode is contract between an application and a printer. It does not matter that there are multiple ways in Unicode to print the same glyph on a printer. The printer does not care! But the DNS cares. The DNS has no concept of "what it means" and alternate Unicode strings that are presented in an identical way on a screen actually

map to distinct DNS names. So, the effort in the use of Unicode in the DNS has been one of trying to push the Unicode glyph set back into the box and try and specify canonical subsets of Unicode that minimise display similarity. This is a tough ask and made even harder by the increasing variance in display glyphs used to display the same Unicode code point. This is most evident in emoji characters. Why is this a problem? Because the Internet still works on a rather crude model of what you see is what you get. If there are alternate ways of coding the same visual outcome, then these are distinct labels in the DNS and can be associated with distinct service points. The possibilities to dupe unsuspecting users is of course a natural and inevitable outcome of this process.

These days "DNS Abuse" is a current topic, particularly in the ICANN world. The phrase describes an effort to engender a level of self-regulation in the DNS supply industry, where behaviours are governed largely by contractual provisions between the registrant and the registrar, and between the registrar and a common registry. It allows various forms of abusive behaviours, including criminal activities, that leverage the DNS to be sanctioned by contractual enforcement including takedown of the DNS names. It's a lot like the self-regulatory measures that are common in the finance industry, but without the reporting framework, without any common legal framework for enforcement and without any penalties for breaches. My suspicion is that it will turn out to be no more effective than the similar measures to undertake self-regulation in the finance sector, and probably even more ineffectual that the rather unimpressive results that the banking sector has posted. It's unlikely to be successful in reducing the levels of abusive and criminal behaviour that leverage the DNS and the Internet.

DNS Fragmentation is also a perennial topic in the evolution of the name space. The pressures for a single consistent name space are embedded in the concept of a single network. Communications systems rely on assumptions of referential integrity, and referential integrity typically implies that the same DNS name refers to the same resource. We've seen this concept being tested many times, from alternate root systems of a couple of decades ago through to private name spaces today in the enterprise environment. A good case in point about fragmentation is the use of search terms as a replacement for the DNS. The objective of a search engine is to try and customise the responses to best match the known preferences of the querier, and when I attempt to pass a pointer to you about a digital resource then the search term I use that will expose this resource may not be exposed when you enter the same search term in your context. However, there are other forms of name fragmentation that are enabled by DoH. DoH enables a name space to be lifted out of common network infrastructure and placed into the context of an attribute of an application. The application can direct DoH queries to server infrastructures of its own choosing and provide responses that pertain to the application as distinct from a lookup in a common distributed database. The ability in HTTPS to push objects to the application client also allows the use of so-called resolverless DNS where an application can improve the performance of name resolution functions by performing them in advance of the time when it is needed.

DNS "name flattening" has been a constant pressure in the DNS. Noone wants to have their critical service names to be buried.deep.down.in.the.dns.under.a.bunch.of.other.names. Not only do such names take longer to resolve, they increase the set of dependencies in the same way as there are presumably a greater number of service providers ideas all the way down in the name hierarchy. DNS users want shorter names. The shorter the better. The result is that the name space is under constant erosive pressure to flatten down. The ultimate place to land is in the top level of the DNS, in the root zone and as the price premium for top level domain comes down the pressure to inflate this zone with significantly larger number of entries is an inevitable consequence.

But perhaps the forces of evolutionary pressure are more fundamental and parallel the evolutionary forces of the Internet itself. The silicon industry is indeed prodigious, and there are many more processors in this world than people. While we have constructed the DNS name space using an analogue of natural language terms as a means of facilitation of human use, this is not necessarily the dominant use pattern of the DNS any longer. One view of the DNS today is a universal signalling and tunnelling protocol, and the use of the DNS as a command and control channel for bot armies testifies to the efficacy of such use

of the DNS! It's likely that as the number of such devices increases then the use of the DNS as a command and control mechanism increases in importance and the human use becomes increasingly marginalised. Human use DNS may well become an esoteric luxury goods business. The high touch activity of DNS name management is unsustainable in a shift from human to largely automated use, and the business models and institutions that populate this space will need to adjust to a names business that provides names not as a branding attribute but as an undistinguished commodity activity. In the same way that we have transformed IP addresses from end point identifiers to ephemeral session tokens, we may well see the DNS as a code base for command and control of highly distributed automated systems, which is a far cry from the distributed database lookup that we originally constructed for the human use model of the DNS. When we think of a DNS query as a set of instructions to a DNS server, and the DNS server as a distributed processing environment, then the DNS changes from a distributed database to a distributed computation and signalling environment. The composition of labels in such a DNS is no longer roughly derived from dictionaries of known words, but instead are encoded instructions where the labels is in effect a program for a name resolver to execute. It is certainly a different future for the DNS as we knew, but it's probable commoditisation in the future is in line with the plight of carriage, switching and content in the Internet! From this perspective the evolution of the DNS parallels the larger evolution of the Internet itself, where the infrastructure is not about a human-usable framework any longer, but instead is focussed on providing a highly automated environment where the elements are themselves programs and automata.

That does not mean that the human use DNS will disappear, but the DNS as we know it today may end up as a small set of high end luxury boutique activities that make a feature of the luxury of custom procedures to manage persistent names, while the rest of the DNS heads deeper into a commodity utility world of large sets of algorithmically generated transient names that are managed entirely automatically and tailored for one-off use by other processes. It may be that the overwhelming use of tomorrow's DNS has nothing much to do with human names any longer and will be concentrated on serving a largely automated framework that leverages the DNS to support a general command and control signalling framework. Ephemeral names are as good as, if not better than, persistent names. Registration and attribution processes are largely irrelevant. The DNS may still be valuable, but individual names would be completely worthless!

Disclaimer

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

Author

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

www.potaroo.net