May 2024
Geoff Huston

Calling Time on DNSSEC?

There have been quite a few Internet technologies which have not been enthusiastically adopted from the outset. In many cases the technology has been quietly discarded in favour of the next innovation, but in some cases the technology just refuses to go away and sits in a protracted state of partial adoption. In some cases, this has in determinate state been so protracted that much of the original rationale for the technology has been overtaken by events and the case to support adoption needs to be rephrased in more recent terms.

> IPv6 is a good case in point where the basic architecture of the protocol, namely as an end-to-end address-based datagram architecture, has become an imperfect fit for a client-server network that makes extensive use of replicated service delivery platforms. Today's network is undertaking a transformation to a name-based network, and running out of addresses to the extent that it is no longer possibly to uniquely address every attached client is no longer the catastrophic event that we once thought it would be. We appear to have attached some 30B devices in today's Internet, yet in terms of IPv4 use, we have achieved this using a little over 3B unique IPv4 addresses visible in the routing system.

In this case I'm referring to secured DNS, or DNSSEC, which has been tied up in progressive adoption for some 30 years. Over this time, we've seen many theories appear as why the pace of adoption of DNSSEC has been so lacklustre, including a lack of awareness, poor tooling, inability to automate operational management, too much operational complexity and a general inability to sustain a case that the incremental benefits of adoption of DNSSEC far outweigh the increased operational costs and added service fragility. Through the lack of clear signals of general adoption of DNSSEC over three decades, then is it time to acknowledge that DNSSEC is just not going anywhere? Is it time to call it a day for DNSSEC and just move on?

Now admittedly this is an extreme position, and I admit to deliberately being somewhat provocative in asking this question to get your attention but there is a grain of an uncomfortable truth here. As a collection of service operators, we appear not to care sufficiently to invest in supporting the additional costs to operate a DNSSEC-secured DNS. After some 30 years of living with a largely insecure DNS infrastructure, we appear to be comfortable with this outcome.

How have we got to this point?

Embarking on the DNSSEC Journey

It was back in 1990 that a researcher reported that the DNS was vulnerable to cache poisoning attacks. What that meant was that an attacker could implant false name-to-address mapping information into a DNS resolver that the resolver would then hold in its local cache and then use it to answer subsequent queries. At the time this was a significant problem for the network because the trust framework at the time was based on the integrity of the mapping of DNS names to IP addresses (DNS resolution) and the

integrity of the routing system in directing IP packets to the "right" destination. To put it simply, as long as you sent packets to the right IP address you necessarily trusted the response as authentic. This DNS attack corrupted the mapping of a DNS name to the corresponding IP address in a way that was impossible for an end user to detect.

The response within the Internet Engineering Task Force was to work on a security framework for DNSSEC:

> "Security is provided by associating with each item of information in DNS a cryptographically generated digital signature. Commonly, there will be a single RSA key that signs for an entire zone. If the resolver reliably learns the public RSA key of the zone, it can verify that all the data read was properly authorized and is reasonably current."
> draft-ietf-dnssec-secext-00.txt, February 1994.

The specification was refined within the IETF process over the ensuring couple of years and published as a proposed standard, RFC 2065, in January 1997.

And then nothing happened.

Why?

Well as the RFC itself says:

> "It should be noted that, at most, these extensions guarantee the validity of resource records, including KEY resource records, retrieved from the DNS. They do not magically solve other security problems. For example, using secure DNS you can have high confidence in the IP address you retrieve for a host name; however, this does not stop someone for substituting an unauthorized host at that address or capturing packets sent to that address and falsely responding with packets apparently from that address. Any reasonably complete security system will require the protection of many additional facets of the Internet."
> RFC 2065, January 1997

The Rise of SSL

In the meantime, Netscape was working on the same generic problem, but from a different angle. Their effort, called the "Secure Socket Layer", or SSL, was released as version 2.0 in 1995. (The earlier version, 1.0, was susceptible to replay attacks). The question that SSL attempted to answer was subtly different to the question being asked by the DNSSEC effort. Here, the question was: "Can the service that I have connected to demonstrate that it is an authentic instance of the service named by a DNS name?"

The security question answered by a successful SSL connection was not necessarily tied to an IP address, and it did not matter what IP address was used for the service end of the connection. As long as the service could demonstrate that it was an authentic instance of the name service then the connection was considered to be authentic.

Their approach used a more conventional application of X.509 public key certificates, using a collection of trusted intermediaries (*certificate authorities*, or CA's) to vouch for the authenticity of the binding between a service operator's public/private key pair and a given DNS name. This approach completely bypassed the need to authenticate the name-to-address mapping function of the DNS and asked the more direct question: "Have my packets ended up at the named service point where I intended them to go?"

The use of SSL (subsequently called Transport Layer Security (TLS) in further revisions when the IETF took over maintenance of this protocol in 1999) has certainly flourished. It has not been without its issues, including a string of security flaws in the protocol, a period of exorbitant pricing for the issuance of domain name x.509 certificates and a steady stream of errors and mishaps in certificate issuance by some of the trusted certificate authorities. However, the tightly knit community of browser developers

and web content server platform vendors were able to maintain coherence of the framework. It was able to withstand the scaling pressures from the transformation of secured content from an expensive luxury service to a freely available commodity through the use of free domain name certificates. These days the penetration of TLS in the world of web content is over 97% of all web requests according to one popular web content platform (https://radar.cloudflare.com/adoption-and-usage).

In contrast, the adoption of DNSSEC has been somewhat lacklustre at best. Some twenty years after the publication of RFC 4033, the stable specification of DNSSEC, we are still asking the question: "Why hasn't DNSSEC been widely adopted?"

DNSSEC vs TLS

It's not that TLS and DNSSEC are directly substitutable, given that they perform different trust functions. It's that TLS actually provides a more useful trust outcome. If the remote service point can prove its bona fides to the connecting client that it is in fact an instance of the named service that the client intended to connect to, then the IP address used by the client to direct packets to this server is not all that relevant.

TLS has few external dependences. A service operator needs to meet the certificate issuance criteria of any of the trusted certificate authorities, which typically means that it must demonstrate that it has control of the associated DNS zone by adding a one-off token to the DNS or performing some similar "proof-of-control" test. The certificate authority will issue a certificate and hand it to the applicant. The applicant will use this public key certificate in the initial TLS exchange, together with performing some function using its private key that the client can verify using the public key provided in the certificate. This functions in a reliable manner as long as the certificate authority is trusted by the client, and the client can support one of the cryptographic functions listed in the TLS exchange.

Wildcard TLS certificates operate in a manner that is subtly different from wildcards in the DNS. DNS wildcards encompass an arbitrary number of levels of hierarchy, so a name such as `a.b.c.foo.example.com` would match a wildcard entry of `*.example.com`. Wildcard Domain Name certificates can only encompass one layer, so only `foo.example.com` would be matched against a wildcard certificate for the name `*.example.com`. In contrast DNSSEC matches the same scope as the DNS scope of wildcards.

We've had both technologies for some three decades. If the level of use is a yardstick to compare these trust frameworks, then the measurements of use of these technologies differ quite markedly. Figure 1 is taken from the Cloudflare Radar site (https://radar.cloudflare.com/adoption-and-usage) showing the distribution of HTTP requests (access without TLS) and HTTPS requests (access with TLS).

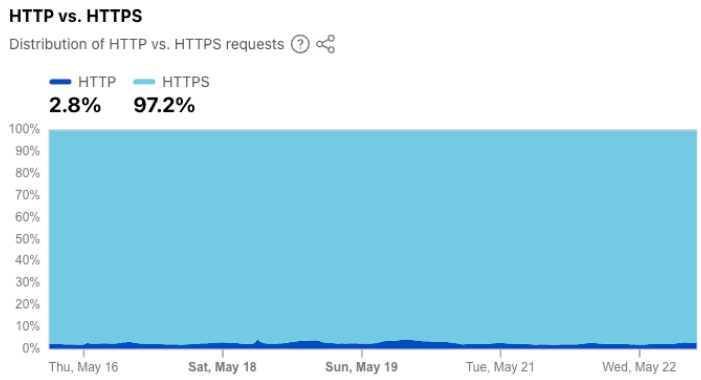

Figure 1 – TLS adoption (from Cloudflare Radar)

How can we measure DNSSEC adoption? We could take the Tranco top 1M domain names (https://tranco-list.eu/) and poll them to see how many are DNSSEC-signed. A recent scan of this domain name list shows that 94,317 of these names are validly signed with DNSSEC, or 9.4%. The issue

with this form of average statistic is that it equally weights all of these million domain names, whereas a more relevant statistic would be to weight each domain name by its relative "use" in some fashion. Perhaps there is a different approach that takes this relative use weighting between domain names into consideration.

One such approach is to look at the ratio of users who sit behind DNSSEC-validating resolvers – this is shown in Figure 2 (https://stats.labs.apnic.net/dnssec/XA) – and then combine this with query data and look at the relative count of queries for domain names that are DNSSEC-signed – shown in Figure 3 (https://stats.labs.apnic.net/cfdnssecdata/).

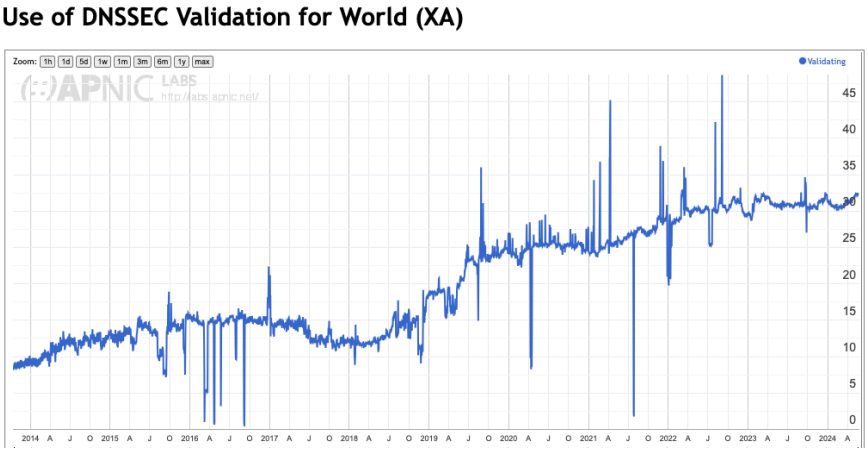

Figure 2 – Proportion of users who use DNSSEC validating resolvers (APNIC data)

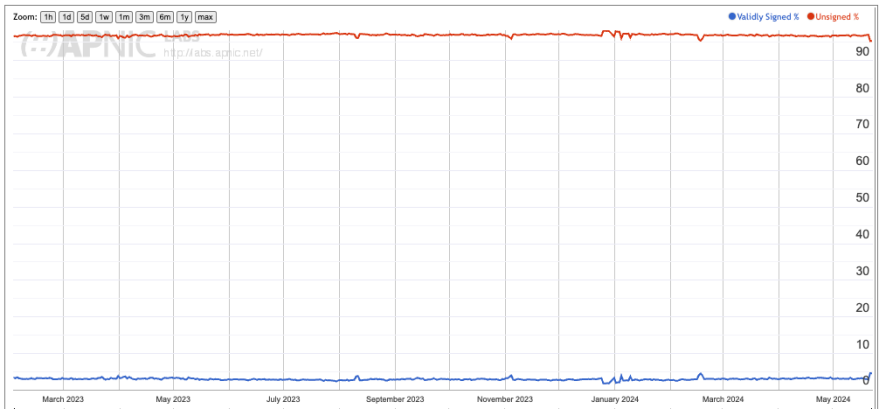

Figure 3 – Proportion of queries for DNSSEC-signed named (using Cloudflare 1.1.1.1 recursive resolver)

The Cloudflare query logs indicate that some 3.2% of queries are made to DNSSEC-signed names, and the APNIC DNSSEC validation data indicates that one third of users are using DNS resolvers that perform DNSSEC validation. Combining these two data sets leads to an estimate that DNSSEC validation is performed around 1% of the time, given the DNS query profile of today's data.

The contrast could not be more stark. The use of TLS is close to universal on today's Internet, while the use of DNSSEC is less than incidental.

Why?

I suspect that the reason behind these radically different outcomes lies in the economics of the Internet rather than the technical merits (or otherwise) of these two approaches.

Economics

The Internet is not organised as a centrally planned technical artefact. The space is largely deregulated, and technologies are used if they create opportunities for providers. Such opportunities may reduce the cost of the production of goods and services or increase the utility of the service for consumers.

How does a security framework fit into such a picture? The major aspect of this question is a cost and benefit exercise. If the incremental cost of deployment of the security mechanism is lower than the perception of estimated benefit of having the security measurement, then the case in support of deployment is far stronger than if the opposite were the case. Superficially both TLS and DNSSEC offer a similar benefit, in that these technologies are intended to provide assurance that the user's packets are being directed to correct destination address. In other words, assurance you are connecting to the service that you intended to use.

We have already noted that the assurances are somewhat different between TLS and DNSSEC, in that TLS attempts to verify that the service is providing assurance that it is an instance of that **named** service, while DNSSEC provides assurance that user is sending packets to the **address** of the service name.

However, the more critical distinction is that DNSSEC is an instance of an **infrastructure** service where the assurance is provided as an attribute of the network's connectivity service, while TLS is an instance of an **application** service, where the assurance is provided by the application.

Services provided by the application have a direct relationship between cost and benefit. In the case of TLS, the application incurs a delay in the initial connection, in order to authenticate the identity of the server and establish a secure session context, and has the benefit has a greater level of assurance of the authenticity of the service as well as an encrypted session. In the case of DNSSEC there is a greater distance between cost and benefit. In order to set up usable DNSSEC credentials for a zone all parent zones need to be signed, the DNS infrastructure needs to store both DNS data and the associated digital signature, the DNS name resolution protocol infrastructure needs to support the use of attached DNSSEC signatures, which implies a shift away from the almost universal use of UDP and a larger TCP query load with its own set of performance and cost issues. That is a large set of third parties who incur some level of incremental cost in supporting DNSSEC with little in the way of direct benefit.

The application is not necessarily aware that a response provided by the local DNS stub resolver has been DNSSEC-validated or not. DNSSEC validation is not normally a service provided by the stub DNS resolver, and the recursive resolver is normally used to perform DNSSEC validation. The result is that the application cannot assume that DNSSEC validation has been performed on the DNS response, and if it wants to provide some assurance that the connected server is authentic it must perform its own measures to authenticate the server irrespective of whether the infrastructure service of DNSSEC validation was performed on the service name resolution. This implies that any incremental benefit provided to the application by DNSSEC is not directly visible to the application. In other words, the application is forced to place no value on the DNSSEC validation process and take its own measures to provide such assurance of authenticity.

In situations where cost and benefit are misaligned then there is little motivation to incur the costs of deployment of the service as others accrue the marginal benefit of such actions. Such misalignment generally leads to market failure. And this situation appears to be the issue faced by DNSSEC.

Can we fix this?

I would argue that we cannot fix this situation with DNSSEC as it currently operates while TLS continues to provide a far clearer alignment of cost and benefit. TLS applications are able to validate the authenticity of the service that they are connecting to without a multiplicity of external dependencies. The additional round trip times taken to complete the TLS handshake result are offset by the benefit of improved robustness of the service and securing the session with encryption that DNSSEC alone does not provide.

That is not to say TLS is perfect. Far from it (see https://www.feistyduck.com/ssl-tls-and-pki-history/, for example). There is a collection of trusted CAs, and each TLS client does not know in advance which CA has issued a domain name certificate for a given name. If a single CA is corrupted in some manner, then it can issue a false domain name certificate for any domain name. Automated certificate issuers can be misled as a consequence of domain name hijacking. Certification revocation is poorly handled, so when compromised certificates are being used, then the means to securely inform clients that these certificates are no longer trusted does not generally exist. Such weaknesses in the Web PKI framework have proved challenging to address over the years, but the lack of viable alternatives that provide similar assurance to an application with a similar alignment of cost and benefit (and I include DNSSEC in this observation) has meant that we continue to rely on the Web PKI and TLS.

The most promising effort to address the shortcomings in the Web PKI framework and also create a new benefit proposition for DNSSEC came with the IETF effort related to placing TLS public keys into the DNS (DANE, RFC 6698, August 2012) over a decade ago. The overall approach taken by DANE was compellingly simple. Rather than relying on a trusted third-party commentary (an X.509 certificate issued by a CA) to associate a public/private key pair with a given domain name, the public part of the key pair could be placed in the DNS zone for that domain name. Rather than relying on a public key certificate and a trusted CA to set up a TLS session, the TLS client could query the DNS for a singled record of the public key and perform DNSSEC validation to verify the authenticity and currency of the key. Some years later a refinement to this DNS-based approach was proposed to improve the speed of the TLS key validation process. This refinement (RFC 9102, August 2021) proposed adding the DNSSEC validation material into the TLS handshake, so that the public key, a digital signature across that key, and the DNSSEC validation path to verify that key was provided in the TLS handshake. A single trust point, the DNS Root key, replaced the collection of trusted CA's.

If this approach had been proposed in the mid-1990's it is possible that DANE could've eclipsed the use of the Web PKI in SSL/TLS. But by 2021 the position of the Web PKI and the associated collected or trusted CAs were so deeply entrenched in the use of TLS that the costs of moving to DANE to store these public keys were regarded as too high. So DANE was unable to gather any momentum of deployment.

Wither DNSSEC?

Where does this leave DNSSEC?

There is no doubt that the DNS has evolved to assume a more prominent role in the architecture of the Internet. What was once a simple name-to-address mapping has evolved in many directions all at once. The service binding work has allowed the DNS to become a service rendezvous mechanism, specifying to clients where and how a service can be accessed. It's a signalling mechanism for verification of a remote party's credentials in domain name terms, and it appears to provide the command-and-control channel for distributed systems (both for good and evil!). Given the overall shift within the network from an address-based architecture to a name-based architecture, then it would seem to be only logical that the case to add verification to the DNS name resolution process would be strengthened over time.

However, the DNS name resolution process was designed to be as efficient as possible. The protocol chose to use UDP as its transport protocol to allow servers and clients to operate in a stateless mode. Subsequent experience has pointed to the effectiveness of this design decision, as long as the DNS payload comfortably fits with a UDP payload without triggering IP packet fragmentation. Substantive issues can arise when digital signatures are added to the picture, and DNSSEC is navigating a difficult path between UDP fragmentation, the additional delays involved in triggering TCP re-query, and the further delays involved in assembling a signature validation path. DNS resolvers are evidently not enthusiastic to deploy DNSSEC validation under such circumstances, and DNS name servers are equally unenthusiastic to enable DNSSEC signing of the zones that they serve.

Some of the more recent extensions to DNS functionality appear to make opportunistic use of the presence of DNSSEC to make them more resilient to various external attempts at manipulation. The Server Name Indication (SNI) field of the TLS handshake, and the more general Client Hello in TLS provide a cleartext indication of the site a user is accessing. Obscuring this SNI information via encryption is certainly desirable from a privacy perspective, but communicating the public key used to encrypt this part of the initial TLS handshake in a manner that resists external efforts to tamper with the value is a challenge. DNSSEC would be the obvious response where the authenticity and currency of the key can be verified by the client, but the rather poor state of DNSSEC deployment, both in signing zones and validating signed names, makes this a piecemeal response. Some would argue that layering additional functions and services on a DNSSEC base further strengthens the motivation for broader DNSSEC adoption. Others would argue that proposing a general Internet service that relies on the use of DNSSEC compromises the prospects of such a service's adoption. The working draft that specifies TLS Encrypted Client Hello (draft-ietf-tls-esni) goes to some effort to make the case that there are alternative ways to improve the tamper resistance of the value of the encryption key that do not necessarily involve DNSSEC.

If the entire case for DNSSEC adoption is based on a generic proposition that a secure and verifiable DNS system is *better* in some abstract sense than an unsecured DNS, then the evidence of adoption of DNSSEC accrued over the last three decades leads to a conclusion that it has not proved to be a compelling proposition for operators of DNS infrastructure. The most heavily used domain names, operated by Google, Facebook, Akamai, Amazon, Microsoft, Apple and TikTok as part of their content hosting products are all unsigned (https://stats.labs.apnic.net/cfdnssecdata/?w=1&a=0) and there appears to be no visible trends in the DNS space that will alter this picture in the near future. Yes, there are benefits that result from fielding a DNSSEC-signed domain name for name publishers, and benefits for clients in performing DNSSEC-validation of signed names, but for most service operators their current evaluation of incremental costs and benefits of DNSSEC simply do not come out in favour of DNSSEC adoption.

We are left with the somewhat frustrating conclusion that the protracted effort to convince the industry that a secured name infrastructure is worth the incremental cost of adoption has not managed to achieve overwhelming success. Progress along these lines is frustratingly slow, and it leads to some more fundamental questions. In a market-based service environment, is infrastructure security a *market failure*? If indeed this is a case of market failure, then is there a case for regulatory interest to be expressed, such as the FCC's recent Notice of Proposed Rulemaking in the US relating to routing security? (https://docs.fcc.gov/public/attachments/DOC-402579A1.pdf)? Are we looking for solutions in the wrong place? If the Internet is increasingly application-centric then what's wrong with authenticity tests that can be applied by the application rather than in the infrastructure itself?

One could argue that the true innovation of the Internet Protocol was not in the IP header, but in the Transmission Control Protocol. Rather than trying to engineer lossless reliable datalink services for networks, TCP embraced a model of unreliable lossy datalinks service and repaired the damage at the transport layer. Is there a lesson to be learned here for DNSSEC design?

Progressing this line of thought, how could we re-think DNSSEC? One option is to dispense with whole-of-zone signing in advance and use font-end DNSSEC signers. We could dispense with the complications of NSEC and NSEC3 records and use compact denial of existence instead (draft-ietf-dnsop-compact-denial-of-existence), removing a source of complexity and DNS response bloat. Could we contemplate turning off the DNSSEC OK flag, and only permit DNSSEC queries in conjunction with TCP transport? Could these front-end signers perform DNSSEC validation and bundle the DNSSEC validation path with the signature in the style of CHAIN queries (RFC 7901)? Could we consider outsourcing the operation of these DNSSEC front-end signers to specialized DNSSEC providers?

What appears to be very clear (to me at any rate!) is that DNSSEC as we know it today is just not going anywhere. It's too complex, too fragile and just too slow to use for the majority of services and their users. Some value its benefits highly enough that they are prepared to live with its shortcomings, but

that's not the case for the overall majority of name holders and for the majority of users, and no amount of passionate exhortations about DNSSEC will change this. In its current form we could do ourselves a favour and just drop DNSSEC and move on. But it feels wrong to just walk away from the objective of securing the name space.

I guess the question we should be asking is: If we want a secured named space what aspects of should we change about the way DNSSEC is used to make it simpler, faster and more robust?

Disclaimer

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

Author

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

www.potaroo.net