**The ISP Column**
*A column on various things Internet*

August 2018
Geoff Huston

# DNSSEC and DNS over TLS

The APNIC Blog has recently published a very interesting article by Willem Toorop of NLnet Labs on the relationship between Security Extensions for the DNS (DNSSEC) and DNS over Transport Layer Security (https://blog.apnic.net/2018/08/17/sunrise-dns-over-tls-sunset-dnssec/). Willem is probably being deliberately provocative in claiming that "DoT could realistically become a viable replacement for DNSSEC." If provoking a reaction was indeed Willem's intention, then he has succeeded for me, as it has prompted this reaction.

In this article I'd like to look at the roles of Security Extensions for the DNS (DNSSEC) and DNS over Transport Layer Security (DoT) and question DoT could conceivably replace DNSSEC in the DNS. Firstly, let's look at what each of these technologies are attempting to achieve in trying th secure the DNS.

## DNSSEC

DNSSEC adds digital signatures to DNS data. The addition of this digital signature is intended to allow a client of the DNS name resolution service to assure themselves of a number of qualities about the DNS response they have received.

Firstly, in a manner analogous to the intent of a a handwritten signature on a document, the client can be assured that a validated digital signature attached to a DNS response implies that the DNS response is the DNS data that the original zone administrator had entered into DNS zone. It has not been altered with or tampered with in any way. The response is "genuine".

Secondly, the client can be assured that the data is current. The date in the digital signature is not enough to ensure that the received information is the absolute latest information that is in the authoritative version of the DNS zone. There may be later information that has been published, but not fully propagated within the DNS, as is the case with the DNS with or without DNSSEC. At the same time the expiration date of the digital signature ensures that the receiver is not being passed a replay of stale information that was originally published earlier than a date specified by the original zone administrator. In other words, DNSSEC can assure the client that this is not a replay of stale information.

Thirdly, while DNSSEC allows a client to detect efforts to alter DNS data, it will not allow a client to correct the alteration. Failure to validate the digital signature of a signed response does not say how the response may have been altered, nor can it indicate what the 'correct' response may necessarily be, but validation failure is intended to clearly indicate that the response is not genuine, and therefore not trustable.

DNSSEC is a set of extension to the DNS, and it does not fundamentally change the DNS. A zone administrator adds digital signatures to the contents of a zone file by adding additional information into the zone through the use of DNSSEC-related Resource Record types (RRTypes). A security-aware DNS resolver uses the digital signature that is attached to the DNS response of the queried DNS name (the 'RRSIG' RRSet of the DNS name), and then retrieves the sequence of in-zone DNS signing key information (DNSKEY RRSets) and DNSSEC Delegation records (DS RRSets) of the zone's parent delegation hierarchy in order to

assemble an interlocking chain of keys that link tor Root Zone's Key-Signing Key to the Zone Signing Key that was used to construct the RRSIG value, in an analogous manner of a validation path in conventional Public Key Infrastructure (PKI) hierarchies. These are all 'standard' DNS queries and responses. Put simply DNSSEC alters what you can ask from the DNS and augments the answers in what you get from the DNS, but not the way in which DNS questions and responses are passed over the Internet.

### What can DNSSEC Protect?

The objective of DNSSEC is to allow a client of the DNS to assure itself that the information that is passed back as an answer to a DNS query is an accurate and faithful copy of the information held in the authoritative servers. It should be able to do so irrespective of whether the response was assembled from a locally cached copy of the information or retrieved directly from an authoritative server for the DNS zone.

DNSSEC does not protect or restrict how the DNS query is handled. When you consider that a general performance objective of the DNS is to serve answers from local caches to the greatest extent possible, the identity of the server that is providing the data is not important. Whether the server is a recursive resolver, a non-authoritative server or any other source of DNS information, is not important to the DNSSEC validation process. Once DNS data is digitally signed, the identity of the agent that provides the DNS response is not relevant, or even the way in which the client has obtained the DNS response is not relevant. What matters is that the receiver can be assured that the data itself is authentic.

> The only minor exception to this general observation that how the answer was obtained is unimportant in DNSSEC is indirection. Where the DNS has redirection pointers, such as CNAME, DNAME, NAPTR, MX, SRV and LUA records, the redirection record itself needs to be validated by DNSSEC along with the final answer. This additional validation is required to authenticate the logical connection between the name in the query to the name and the response that has been formed by this indirection.

DNSSEC can protect against various forms of manipulation of DNSSEC-signed data. Where responses are altered the digital signatures will not validate, and clients can detect this mismatch as a validation failure. Where the middleware deliberately withholds data, including withholding the DNSSEC signature structure, DNSSEC validation can reveal this effort to withhold data.

However, being able to identify that data has been altered and preventing the alteration in the first place are slightly different objectives. DNSSEC cannot prevent data manipulation of DNS responses, nor can it inform a client as to what the authentic response should have been. DNSSEC does not encrypt DNS data. An observer can still look at DNS activity, irrespective of whether the data is DNSSEC-signed or not.

### DNS over TLS

DoT wraps up a DNS protocol transaction within an encrypted channel. When a sender places information into a TLS-protected channel the data that arrives at the receiver is precisely the same data that was passed into the channel.

As well as channel protection, TLS offers some level of authentication of the remote party. In setting up a TLS connection the remote end demonstrates its knowledge of a private key that is associated with the DNS named identity of the remote server. The connecting client has also been provided the matching public key by the TLS handshake, and an attestation made by a trusted source that this public key corresponds to the named identity of the remote party.

> For example, "Let's Encrypt attests that you can trust that this public key is associated with the service named "www.potaroo.net", because the TLS

handshake provides the client with a public key certificate for www.potaroo.net signed by Let's Encrypt."

If the server takes a token and then encrypts this token using the server's private key, then the client can use the corresponding public key to reveal the original token value.

This implies that irrespective of the IP address of the service, the service is demonstrating that it is an instance of the named service by demonstrating possession of the private key that matches the public key contained in a trusted public key certificate.

DoT is not a datagram service. This is an important distinction because DoT uses a TCP transport as the basic connection protocol and layers TLS encryption and session integrity and then carries DNS over this connection. This means that DoT can avoid the issues associated with IP level fragmentation that occur with DNS over UDP and be able to carry larger DNS payloads. This distinction is also applicable when comparing DoT with Datagram Transport Layer Security (DTLS) (RFC6347), as DTLS is intolerant of IP level fragmentation.

### What can DoT Protect?

The DNS is a form of a hop-by-hop protocol. When a client asks the DNS to resolve a DNS name it does not necessarily direct the name resolution request to the authoritative name servers for that domain name. Instead, the client can simply ask its local recursive resolver. If this server already has a locally cached copy of the response, then the recursive resolver will provide the answer directly. The server will only generate a corresponding DNS query if the name is not contained in its local cache. The target of this query may be an authoritative server for the zone in question, or it may be another recursive resolver of some shape or form.

Answers are also treated in a hop-by-hop manner. The authoritative server will pass DNS data to a set of visible resolvers in response to queries. These resolvers, in turn, may pass DNS data to other resolvers, and so on, until data is passed to an end client. There is no requirement for synchronicity in this process. A caching recursive resolver will only query for a name when its locally cached copy has been held for the requisite local cache holding time. The critical observation is that there is no requirement for a simultaneous set of paired connection states between each of these resolvers. Equally, there is no assurance that a virtual path from authoritative server to client exists to pass the DNS response. The entirety of the DNS can be seen as a set of pairwise operations where a server sends a response to a client.

TLS is a session protocol and is intended to allow a client and server to exchange data with a high degree of confidence that the data is not being altered or manipulated during transit between the sender and the receiver. In a hop-by-hop information propagation model TLS can protect each instance of the network hop, but not necessarily protect the entire sequence of hops.

But even that limited level of protection may be enough. In today's network the protection of the session appears to be a fundamental issue. Many regulatory frameworks that attempt to restrict the content available to users, or monitor user activity for various reasons, do so by intervening in the DNS query stream. Blocking access to a site can be as simple as intercepting all DNS queries and responding with incorrect information for queries relating to the names of blocked resources or servers. This technique relies on some form of network middleware intercepting all packets directed to remote port 53 and redirecting these packets to a local recursive resolver that is configured to operate promiscuously and has been configured with the applicable filtering rules.

The use of DoT prevents this form of DNS interception. When the TLS session is opened the remote end has to demonstrate to the local client that it has possession of the private key associated with the name of the server. Once the session has been set up the middle unit has to intrude upon the encrypted conversation if it wishes to alter DNS responses. This is intentionally a significant challenge.

Also, as the session is encrypted, any man-in-the-middle observer attempting to see the DNS transactions on the path between the client and the server will be unable to observe the content of the DNS queries or responses.

What TLS cannot protect is the transitive information flow from one hopp to the next. If a client establishes a DoT session to a recursive name server, then it can be assured that the responses it receives are the responses that the recursive name server sent to the client. What it cannot know is whether these responses are a faithful and accurate copy of the information that was originally provided to the recursive name server, or whether the recursive resolver itself is performing some unauthorized manipulation of the DNS information.

This leaves TLS vulnerable to at least three forms of attack. The first is crude disruption, where the attacker intrudes on the data path and discards or simply alters the payload of all TLS packets. The second is subtler and requires the attacker to subvert the client's recursive name servers and corrupt their local DNS caches and pass false information along the protected session. The third is also subtle and involves an attack on the domain name Public Key Infrastructure. If an attacker can subvert a trusted Certificate Authority directs it to issue a fake domain name certificate for the server. This then allows the attacker to spoof the identity of the server and the client will incorrectly assume that the TLS session is terminated at the trusted server whereas in fact the attacker is now feeding information to the client.

## What if Everyone Did This?

Digital signatures cannot certify what is false. They can only certify accurate and complete copies of original data. In looking for false information you either have to assume that everything is correctly digitally signed all of the time, and the absence of a digital signature or the inability to verify a digital signature is a reliable indication of untrusted data. The alternative is to have verifiable meta-information that indicates what data is comprehensively digitally signed, and what is not. The same assumption, that the absence of digital credentials when such credentials are expected, or credentials that cannot validate, can be used as reliable indicators of falsified data.

While universal deployment of a security mechanism is a desirable assumption, it is rarely achieved. The practical consideration is that these security mechanisms must operate usefully in an environment of partial deployment, and that requires a robust verifiable mechanism of declaring in advance what is protected by digital credentials.

From such a perspective the suppositions in Willem's article that imagine the universal deployment of TLS break down. A client may have a secure TLS connection to a local recursive resolver, but that resolver may use a non-secured channel to reach its DNS servers. How is the client to know that while the data is being passed reliably from the recursive resolver to the client, the data itself may not have been reliably gathered by the recursive resolver? DoT simply can't help here. Equally, DoT does not sign the content, so if fake information is being passed along the secure channel DoT just can't help.

## Protecting Content and Carriage

It appears to me that DNSSEC and DoT address distinct security objectives and neither technology can realistically replace the other.

Protection of the integrity of content is necessary in a distributed system. Irrespective of how the DNS data is obtained it is necessary for a DNS client to assure itself that the data is authentic and is a faithful and complete copy of the original authoritative data. DNSSEC can provide this assurance and do so in an environment that also supports partial deployment.

Protection of the channel provides essential assurances to end clients. It can assure them that their own activities are not exposed man-in-the-middle observation and is allows them to be assured that the DNS server that they think they are communicating with is indeed the server that they intended.

Almost.

This is not quite the full story.

There is still the open problem of the weakness in the widespread trust of the current domain name PKI. Trusting more than 1,500 Certificate Authorities to always operate to the highest levels of integrity is such a lofty objective that it stretches rational credibility into the realm of credulity. Contrary to the assertions of many PKI apologists, Certificate Transparency does not stop the problem. At best, Certificate Transparency might inform you that a problem occurred in the past. It seems to me that DANE (Domain Name Keys in the DNS) really does play an essential role here, and using DANE and DNSSEC to reliably link a domain name to a public key seems to me to be the only effective response we know of to address the threat of rogue or corrupted CAs.

There is also the difference between DNSSEC in theory and DNSSEC as we see it working today. In theory, a security-aware DNS client should not trust a recursive resolver's DNSSEC validation outcome. It seems that the current practice is that clients rely on a security-aware recursive resolver to withhold the response and instead return a SERVFAIL response code if the response cannot be validated. If the response is valid then the recursive resolver sets the Authenticated Data (AD) bit in the response. For a man-in-the-middle attack this is ridiculously easy to tamper with as long as the session between the client and the recursive resolver is open. Here DoT helps, in that the man-in-the-middle simply cannot alter the data provided by the recursive resolver. In situations where the client has outsourced DNSSEC validation to the recursive resolver, the DoT improves the overall situation for the client in being able to trust the DNS answers it receives.

However, this still leaves me with a feeling that it's just not quite enough. An appropriately cautious client would not outsource its validation. It should insist on undertaking its own validation of DNS responses. Even if it uses a recursive resolver, it can still perform its own validation of responses. But this additional step takes additional time, and now the entire issue of speed and responsiveness comes into play. It's not only browsers, but for many browsers that speed is everything. Anything that slows down the 'responsiveness' of the browser in responding to a user is frowned upon. When performed in an iterative manner DNSSEC validation can be tediously slow. Similarly, validation of DANE answers can be tediously slow. So, it's fine to insist that user applications should exercise an appropriate level of security paranoia and not outsource their validation, but at the same time its often contradictory to insist that these same applications operate in a highly responsive manner. Maybe DNSSEC itself requires more tweaking, and the concepts explored in RFC7901 of validation chain queries and responses might help. However, in an unexpected manner DoT enters the picture once more at this point. The chain query and response model mandate the use of source-IP-verified transport for large payloads. DoT provides a TCP transport that can handle large responses, prevents spoofing of the client IP address, and allows the client to assure itself of the identity of the server that is providing DNS responses. All necessary attributes when managing this form of chain query and response.

## Conclusions

As should be evident by now, I don't think it's a case that DoT pushes DNSSEC into some lingering sunset afterlife. I think a secure and trustable DNS demands the use DNSSEC irrespective of the fate of DoT.

But DNSSEC is just not enough. There is no doubt that the unencrypted trusting DNS as we use it is widely abused. DNSSEC can illustrate when DNS responses are being manipulated, but they can't stop it. DNS over TLS is not a panacea either, but the TLS channel can at least assure a user that when they pass their DNS queries to a server that they are prepared to trust, then their transactions with this server are private, secure and the server itself is the server that the client intended to use.

It is critically important in a public communications utility that every individual's use of the communications system is private, secure and operates with integrity, and the individual can assure themselves that this is the case. In this respect both DNSSEC and DoT has an important role to play together.

In terms of a secure and trusted DNS, we need more security elements, not less.

## Disclaimer

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

## 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*