September 2018
Geoff Huston

Measuring the KSK Roll

When viewed as a network infrastructure, looks can be very deceiving when looking at the DNS. It appears to be a simple collection of *resolvers* and *servers*. Clients pass their DNS name resolution queries to *resolvers*, who then identify and ask an appropriate authoritative name server to resolve the DNS name, and the result is passed back to the client. Conceptually this look a lot like the model Figure 1.

Figure 1 – A naïve model of the DNS

This may well be the case for some parts of the DNS infrastructure, but the larger picture of how the pieces are put together is more complex. Users often are configured with more than one DNS resolver, and use local timers to repeat resolution requests to other resolvers if a response is not forthcoming quickly enough. To speed up name resolution many DNS resolvers operate a local cache of answers, and where the question is already resident in the cache they can use the cache to respond immediately. Resolvers may use *forwarders* to pass the query on to a resolver that may operate a larger cache. There are also large scale *resolver farms*, where a number of resolvers are linked together to share the query load, and they may also operate across a single cache. Authoritative servers also come in a number of flavors, and these days *anycast* server constellations are a common way to distribute authoritative DNS information across the Internet. What this means is that the DNS infrastructure may look a lot more like Figure 2.

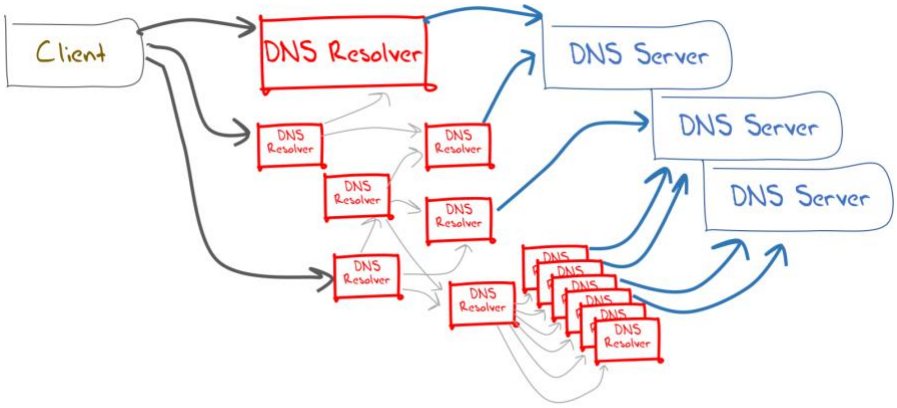

Figure 2 – Variations of DNS Infrastructure

Is it possible to 'follow' a query though this DNS infrastructure? Can we track a DNS query and see its progress through the infrastructure of resolvers and servers to determine how the response is generated? Even this is not as simple as it might sound. A resolver must first establish the network addresses of the set of name servers that are authoritative for the zone that contains the DNS name, and the resolver may not know this information in advance of the query. A simple client query may trigger a resolver to undertake a sequence of queries to discover both the zone *cuts* where the name space has been delegated to a child zone, and the addresses of the authoritative servers for these zones. Can we determine if resolvers used their local cache or whether they

passed a query directly from an authoritative name server? Can we track the response back to the original client? Unfortunately, the DNS protocol provides nothing in the way of assistance here. There are no tracking identifiers in a DNS query, no form of resolver referral count, analogous to the IP hop count, no server 'stamp' on responses and no means to record how a response was generated.

From the client's perspective the DNS operates as an opaque cloud. Queries enter the cloud and most of the time responses come back. The view from an authoritative server is similarly opaque. Queries are passed to the server and responses are generated based on the information in the zone.

There is no clear signal why the query was generated, whether by a client, by a resolver's cache refresh operation or by some misconfigured system that has generated the query in error. This means that most forms of measurement in the DNS are challenging at best. How fast is the DNS? It depends. How reliable is the DNS? Again, it depends. Will using DNSSEC make name resolution slower? It depends.

The KSK Roll

One DNS question in particular has been the topic of much study over the past two years. When we replace the cryptographic key used as the trust anchor for the security framework of the DNS, will there be any disruption to users?

This is a particularly challenging question to try to answer as we cannot generate a test situation in the DNS where name resolution relies on whether or not a particular root zone key-signing key has been loaded into a security-aware resolver's trusted key set for the root zone in advance of the roll of the key.

What do we know?

APNIC DNSSEC Measurement

It's not as if the DNS is completely opaque, and there are some aspects of DNS resolution infrastructure that can be measured. At APNIC Labs we've been using a large-scale measurement platform for some years now. Originally intended to measure the extent of deployment of IPv6 in the Internet, it is possible to use the same approach to measure aspects of the DNS infrastructure.

The measurement script within an advertisement is restricted to loading of specified URLs. However, this is enough for our purposes as we can load each presented ad with a small set of URLs each using a unique DNS name. The uniqueness of the DNS name ensures that the caches in the DNS resolution infrastructure will not be triggered, and each query made by an end client will generate a corresponding query at the authoritative server for the domain. What if we were to pass two domain names to a client, both of which are DNSSEC-signed, and where one signature is deliberately altered so that it cannot be validated? Users who used DNS resolvers that did not perform DNSSEC validation would retrieve both URLs, while users who used security-aware DNS resolvers would only retrieve the URL of the validly-signed DNS name. In order to remove some ambiguity in the measurement we use a DNS name structure where the zone itself is also a unique name, so that the resolver's DNSSEC queries for the zone's DNSKEY RRset are also passed to the authoritative server.

Using this approach, we can report that some 15% of the Internet's user base pass their DNS queries to DNSSEC-validating resolvers and will not resolve a DNSSEC-signed DNS name if it's signature cannot be validated.

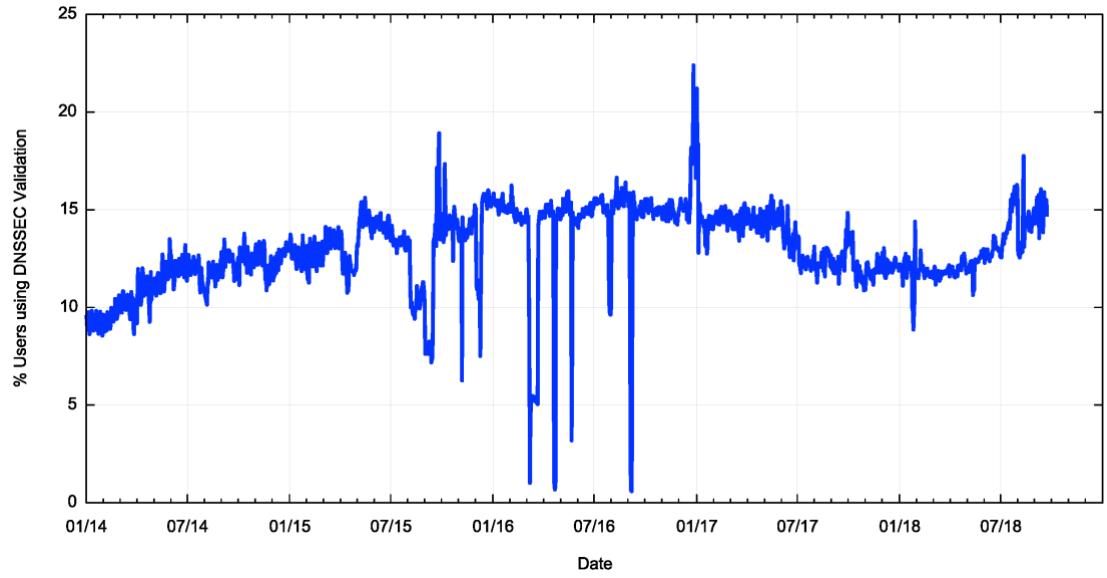

Figure 3 – Time series measurement of users whose resolvers perform DNSSEC Validation

This pool of users is the population that is potentially impacted by a change in the root zone KSK, and we have a reasonable idea of where these users are located, and even which resolvers perform DNSSEC validation.

Measuring the KSK Roll

How can we tell how many of these users may be impacted? And can we tell which resolvers may be causing the problem?

Unfortunately, there is no clear test that can be set up within the DNS name space while the new KSK value is being introduced in the root zone's DNSKEY RRset. If a name is signed with the new KSK, then a resolver that only trusts the old KSK will still validate the name.

> This is because when the new KSK is 'introduced' it is loaded in to the DNSKEY RRset of the root zone. While the new key is being introduced the DNSKEY RRset is signed by the current KSK. Clients will be able to validate a digital signature that is generated using the new KSK because the old key has signed over the new KSK, even if the new KSK has not been loaded into the client's local trusted key set.

If we want to measure the extent of readiness of DNS resolvers to accept an incoming KSK we need to measure the extent to which the new KSK has been loaded into the local trusted set of resolvers, and to do this we need to change the behaviour of resolvers in some manner.

The first effort in this direction was published in April 2017 with RFC 8145. This document describes a method that proposes that DNSSEC-validating resolvers report the trust by periodically sending special "Key Tag queries" to a server authoritative for the zone. There is some level of uncertainty associated with this approach. Querying a root server for a DNS name such as `_ta-19036` will generate an NXDOMAIN response today, and if querier is sitting behind a validating resolver that performs some form of NSEC caching (RFC 8192) or local root zone (RFC 7706), the queries will not make it to the root servers.

The first look at this RFC 8145 query data was surprising. I'll reproduce here one slide from a presentation made to DNS OARC in October 2017 which is illustrative of the issue which has led to the postponement of the key roll (Figure 4).

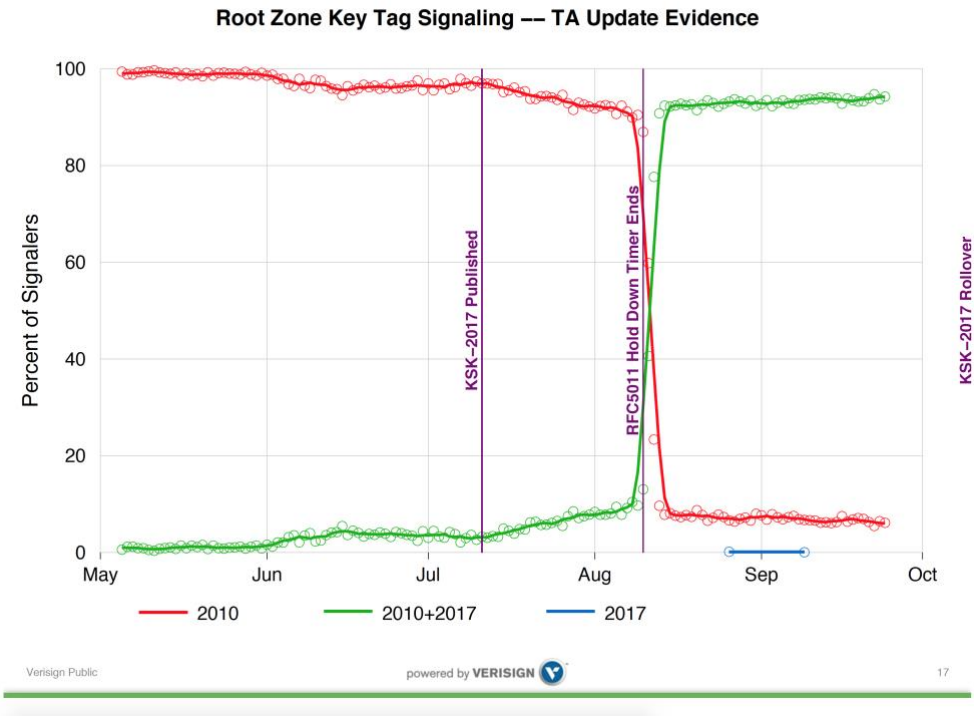

Figure 4 - Trust Anchor Update Evidence – Duane Wessels, DNS OARC 27

The 30 day 'hold down' timer described in RFC5011 appears to have worked as expected. Some 30 days after the introduction of KSK-2017 into the root zone, more than 90% of reporting IP addresses reported that they now had added KSK-2017 to their local trusted key set. A smaller number of reporters had loaded KSK-2017 earlier than the 30 day hold down period, indicating some local configuration, possibly relating to manual key management.

However, there remains a set of resolvers reporting that they still only trust KSK-2010. The report suggests that somewhere between 5% to 10% of reporting IP addresses had not shifted to trust KSK-2017, and in the 40 days since the hold-domain timer expiration this number did not appear to be getting any smaller.

With time running out before the actual key roll, and with some serious concerns about the level of uptake of KSK-2017 from this reported data, then it was entirely reasonable to postpone the key roll. We needed some further time to understand what was happening here and needed to reassess the safety of this action before proceeding further.

Why were there a set of resolvers reporting the continued reliance on KSK-2017?

RFC 8145 Issues

There are some issues with the implementation of this form of Trust Anchor measurement.

The first issue is that there are several false signals intruding into the data set. An implementation of this mechanism permitted resolvers to emit these trusted key state queries even when the resolver is not performing DNSSEC validation. The second is the nature of the query signal. If a resolver uses a forwarder to pass its queries towards the root zone of the DNS then the trusted key queries will still be made, but they will be passed to through the forwarder, and it will appear to the root servers that the forwarding resolver is emitting these queries about its own trust anchor key state. The identity of the original resolver is lost in this case, and the resultant signal is prone to misinterpretation. In addition, this query and its response will be placed in the local cache. This means that there is a potential for multiple resolvers sitting behind a common forwarding resolver to have their trust anchor signals masked, or for the forwarding resolver to make what appears to be

inconsistent signal queries to the root servers. In most cases this masking is not an issue, and the cache lifetime for NXDOMAIN responses is short. However, in the case where the caching resolver is a DNSSEC-validating resolver, and if it is performing RFC 8192-style NSEC caching, then the local cached entry of the non-existence of these 'special' domain names will be longer, and much of the original signal from these 'hidden' resolvers will be masked.

At a more fundamental level there is the mismatch of the signal to the intent. If the original intent was to understand the potential population of affected users, then counting resolvers is just not the same. The distribution of users to the DNS resolvers that they use is heavily skewed, and less than 10,000 individual resolver IP addresses are seen to ask authoritative servers queries on behalf of more than 90% of the entire user population of the Internet. A very heavily used resolver reporting that it has not loaded KSK-2017 is a far more impactful observation than if my local DNS resolver that serves just me sends a similar signal. The inference here is that interpreting the signal provided by these trust anchor queries is difficult unless you can also add some element of 'weight' based on served user population data. However here the second issue, that of forwarders, intertwines. The highly used resolvers are far more likely to be forwarders for other resolvers, and if they pass on a `_ta-19036.` query to a root server, that should not be interpreted as being a reliable signal that this highly used DNS resolver has not loaded KSK-2017 into its trust set, nor even that the reporting resolver even performs DNSSEC validation itself.

The first two cases are these are instances of a 'false positive' signal, where a resolver is incorrectly reporting failure to trust KSK-2017 when in fact it is either not relevant (in the case of non-validating resolvers) or just not true (as the query is actually describing the key state of a hidden resolver), and the third is the problem that there is no clear signal as to its relative 'value' in interpretation.

There are also cases where a resolver will indeed fail to trust KSK-2017, so not all of these queries are in error. As noted already, if a manually managed configuration is waiting right up until October 11, and it is reporting its trust anchor state, then it will correctly report only trust in KSK-2010. On the other hand, there is also the potential for a resolver to be configured in a manner that it cannot follow RFC5011 even though it is configured to do so. Some resolver configuration combinations require the trusted key state to be written to a file store before it is included into the local trust set. If the resolver is operating in an environment where the file store is not writable for any reason (and such reasons exist) then the resolver will be unable to pick up the new trust anchor even though its configuration directs it to do so.

This is a difficult situation. We have a signal about the use of trust anchors that is being generated by an extremely small fraction of all the DNSSEC-validating resolvers out there (and some non-validating resolvers, as it turns out). The lack of clarity of the signal means that while it is clear that caution is now advised, it is entirely unclear as to the conditions for this signal to reliably provide a 'green light' to proceed.

We needed to think about this some more.

KSK Sentinel

We've now seen the description of a different approach to observing the trusted key state of resolvers that reverses the signal flow. Instead of using queries to signal a resolver's trust anchor state this approach uses responses to perform a similar function. A DNSSEC-validating resolver that supports this mechanism will invoke a special behaviour if the left-most label matches a certain pattern.

If the left-most label of a query name matches the pattern `root-key-sentinel-is-ta-<key-tag>`, then if a validating resolver has loaded the key with the given key tag into its trusted key set for the root zone then it will return a 'normal' response, but if the key has not been loaded then it will return a SERVFAIL response to A or AAAA queries that match this pattern.

Similarly, if the left-most label matches the pattern `root-key-sentinel-not-ta-<key-tag>`, then if the resolver has not loaded the key with the given key tag into its trusted set then it will return a 'normal' response, but if

the key has been loaded then it will return a SERVFAIL response to A or AAAA queries that match this pattern. This is the opposite of the "is-ta" behaviour.

This approach can be used to test a single resolver, but it also can be used for sets of resolvers. This is a useful property, because a user will only be impacted by a KSK roll if all of the user's resolvers perform DNSSEC validation and at least one of these resolvers has loaded the new KSK into their trust set.

To measure these, we need to use a minimal set of three DNS tests:

- *an-invalidly-signed-DNS-name.example.com*
- root-key-sentinel-not-ta-19036.*example.com*
- root-key-sentinel-is-ta-20326.*example.com*

If any of the user's DNS resolvers does not perform DNSSEC validation, then the user will be able to resolve the first invalidly signed domain name. This is because the DNSSEC validation signal of SERVFAIL is not necessarily a signal of resolution failure, but a signal that the server has been unable to provide a response for some unspecified reason. The user client's action is to repeat the query using another resolver in the user's resolver set and the user client will only complete the resolution function when a resolver returns a response, or when all resolvers have been queried. This first test is to filter out all clients who use a non-validating DNS resolver in their resolver set.

The next test is to filter out those clients who use resolver sets where not all of the resolvers recognise the key sentinel label and apply special processing. A validating resolver will have the current KSK loaded into its local trust anchor set, and if it recognises the key sentinel label it will return SERVFAIL. This query will only complete when either one resolver does not recognise the key sentinel label, and will return with the response, or when all resolvers recognise the key sentinel label, in which case all resolvers return SERVFAIL. So, this test is to filter out all clients who are have at least one DNS resolver that does not recognise the key sentinel test in their resolver set.

The third test is to determine if at least one of the user's resolvers has loaded the new KSK. It is not necessary to test all of the user's resolvers because as long as one of these resolvers has loaded the new KSK then the user will not be impacted by the KSK roll.

Server Name Indication in TLS

In the ad-based measurement tests we have no direct way to get the ad script to report on a DNS resolution operation. The ad script permits us to perform an HTTPS GET of a web object, but not much more. Let's look at the DNS name of this URL.

We want to achieve two goals here – we want to use unique labels in order to remove any side-effects of DNS and web caching and we want to use domain name certificates to ensure that we can support HTTPS web retrievals, which these days is essential in the online ad environment.

There is a subtle distinction between the DNS and the web environments that makes these two objectives impossible here. While DNS wildcards can be used to encompass any number of labels, a domain name wildcard certificate can only be used for the single left-most label. But for the KSK sentinel test the left-most label is not unique. We need to make the second last label unique, which is fine in the DNS but cannot be achieved in the web for domain name certificates.

But we can embrace failure here and still get what we need. The TLS handshake that starts an HTTPS session sends the server the DNS name of the web server it is trying to connect to as the Server Name Indication field. In the context of this measurement exercise we are using the web retrieval operation only as a signal that the user has received a response from the the DNS. It's not important for the user to actually retrieve the web object, but it is important that the server can see the attempt to retrieve the named object, and the SNI field is enough in this case even though the HTTPS connection attempt will fail due to a lack of a certificate.

The KSK Sentinel Measurement

Even with the use of SNI fields, the residual problem with this form of measurement is that the web fetch is unreliable. The ad may not run to completion and there is a non-negligible count of ads where the DNS query is seen by the ad servers, yet no web fetch is seen. One way to improve the clarity of the signal is to increase the number of tests and in this case we are using the following 6 URLs to form the measurement test:

1. *unsigned-name.example.com*
2. *valid-signed-name.example.com*
3. *invalidly-signed-name.example.com*
4. no-ta-10936.*example.com*
5. is-ta-10936.*example.com*
6. is-ta-20236.*example.com*

We look at those experiments where we see DNS A or AAAA queries for names 1 through 6, and DNSKEY queries for the zones for names 2 through 6. This way we include only those experiments that have commenced to load all 6 URLs by attempting to resolve (and validate) the DNS names.

The only way we can determine if the endpoint has successfully resolved the DNS name is to look to see if the corresponding web object retrieval is attempted. For HTTPS sessions we look at the SNI field and for HTTP sessions we will see the web object request in the server logs. The pattern of web objects gives us the following interpretation:

- **DNSSEC-Validating** - If URL 3 is loaded – the user has at least one resolver that does not perform DNSSEC validation. If URL 3 is not loaded then we presume that all the user's resolvers returned SERVFAIL because of DNSSEC validation.

- **Loaded KSK-2017** - If URLs 1,2,5,6 were loaded, then the user's browsers appear to recognise the KSK sentinel mechanism, and at least one has loaded KSK-2017 into its trust key set for the root zone.

- **NOT Loaded KSK-2017**- If URLs 1,2,5 were loaded then all the user's browsers appear to recognise the KSK sentinel mechanism, and none of these browsers has loaded KSK-2017 into its trust key set for the root zone.

- **NO KSK Support** - If URLs 1,2,4,5,6 were loaded then **a**t least one of the user's browsers does not recognise the KSK sentinel mechanism.

- **Noise** - Any other combination. This is either an abnormal termination of the Ad script before it has run to completion, or a potential indication of a faulty implementation of the KSK sentinel by a resolver.

The first set of results of this measurement are shown in the following table.

Date	Experiments	DNSSEC Validating	Loaded KSK-2017	NOT Loaded KSK-2017	Noise	No KSK
19/9	3,636,908	524,455	20,126	325	20,331	483,672
20/9	3,650,965	545,019	19,802	162	15,170	509,884
21/9	3,542,058	510,565	19,974	492	17,632	472,466
22/9	3,323,320	495,515	21,213	424	13,391	460,486
23/9	3,569,771	509,613	21,757	380	15,472	472,003
24/9	3,458,128	500,069	19,717	413	16,935	463,002

Admittedly, these are early results, but they encompass some 3.5 million measurement tests per day and represent a broad cross section of the Internet's user population.

Some 14.5% of users use DNSSEC-validating resolvers. Of these some 7% - 10% of users are using resolvers that support the KSK sentinel mechanism. Some 4% of users are using resolvers that have loaded KSK-2017, and less than 0.1% of users are using resolvers that are indicating that they have not loaded KSK-2017. This 'not loaded' number is very much lower than the total noise level and is more likely to be attributed to noise in the experiment than a systematic failure to load KSK-2017. The 0.1% figure is therefore more likely to be an upper bound of the number and not a reliable indicator of the extent to which KSK 2017 has not been loaded.

The noise values are worth a further comment. Two thirds of the experiments that have been grouped into the 'Noise' column only retrieved URLs 1 and 2 and retrieved none of the 3 KSK Sentinel test labels. It is possible that this is due to a faulty implementation of the KSK sentinel test that returns SERVFAIL to all of the test labels, which implies that the residual noise levels are down to between 4,000 to 6,000 measurements per day.

Conclusions

It has been a trade-off between waiting long enough to have the key sentinel mechanism deployed in sufficient volume in resolvers to generate statistically valid outcomes and yet start this measurement prior to the planned roll of the KSK on 11th October 2018. These are early results, and reflect less than one week of measurement, but some strong signals are evident in the data.

Of the users who use key sentinel-aware DNS resolvers where we found a clear capability signal, some 99.93% of these users use DNSSEC-validating resolvers where KSK-2017 is part of their trusted key set. The remainder, some 0.07% of these users, were indicating that KSK-2017 had not been loaded as yet.

When considering the larger pool of all users, then these results indicate that some 0.01% of the Internet's entire user population may encounter some DNS resolution issues when the KSK rolls.

The uncertainty levels in this measurement are ± 0.22% as measured by the noise component of the results. As this is far greater than the non-readiness measurement of 0.01% of users, then perhaps there are grounds to believe that the situation may be better than these figures would indicate, and the true level of non-readiness for the KSK roll is far smaller than the 0.01% of users that these results indicate.

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