The ISP Column
A monthly column on things Internet

Geoff Huston
March 2016

Rolling the Root

In the world of public key cryptography, it is often observed that no private key can be a kept as an absolute secret forever. This does not mean that a private key remains a secret for a limited time and then the underlying cryptography spontaneously breaks apart and the key inevitably reveals itself! However, there are evolutionary factors that tend to erode the integrity of a private key over time. The more a key is used to encrypt products the greater the number of clues that are left behind as to its secret value. Of course these clues are infinitesimally small in value and most forms of use of a private key, even relatively intensive use, leave an attacker no better off terms of being able to crack the key, but nevertheless, in absolute terms it's still a risk. At the same time the continuing march of computing capability makes previously impractically difficult computing problems more tractable, so older keys that may have been state of the cryptographic art some years ago may well be more susceptible to modern computing capabilities. And of course there is always the risk of misadventure, so that that the private key is inadvertently revealed, or just as bad in some ways, the private key becomes inaccessible. This latter case does not mean that the private key value is compromised, but the end result is similar – the key is effectively unusable.

DNSSEC and Key Administration Practice Statements

So no cryptographic secret is can be regarded as "absolute". It's all relative and use of a key has some element of associated risk that the key is not a strict secret. If that's the case, and if we cannot really understand the exact levels of risk associated with use of a key, then why should we place our trust in any form of public key cryptography?

To allow us to make a better informed decision about the trustworthiness of a public key it is common convention for the key administrator to publish a "Practice Statement" that details the key holder's intentions as to how the key is to be managed. This is a public commitment by the key administrator on the practices they use to maintain the integrity of the private key.

Typically, one would expect to see in such a document a statement of requirements, a description of the facilities and management controls that apply to the private key, technical controls, operational actions, and intentions for compliance audits. So as long as the key administrator adheres to the commitments in their published Practice Statement, then users who rely in the integrity of the private key as the foundation for their trust in signed objects in the associated Public Key Infrastructure (PKI) have something that is intended to be more substantive than just uncorroborated blind faith!

The DNSSEC key infrastructure for the DNS does not use conventional X.509 public key certificates, but it does use a hierarchical public key structure, where the key signing chains are contained within the name structures hierarchical delegation model. At the apex of the DNS name structure is the root zone, and at the apex of the corresponding public key infrastructure is the Key Signing Key (KSK) of the Root Zone.

If we want to trust in the integrity of this KSK value, as the foundation for trust in DNSSEC, then we need to look to the Practice Statement for the Root Zone KSK Operator. This document was published by the Root DNSSEC Design Team in October 2010, and can be found at the URL: https://www.iana.org/dnssec/icann-dps.txt. Section 6.5 of that document reads:

```
        Each RZ KSK will be scheduled to be rolled over through a key
        ceremony as required, or after 5 years of operation.
```

There has been some debate about the intent of "after 5 years of operation" as to whether this phrase implies a commitment to roll the KSK value on or around the 5th anniversary of operation of the KSK, or literally any time whatsoever after the 5th anniversary, which may include a subsequent lapse of possibly decades or longer! A reasonable interpretation of this commitment is an approximation of the former one, namely that the intent was to roll the KSK on or some time after 5 years of operation. Given that the KSK was introduced into operation in July 2010, that would imply that the KSK roll was intended to occur around mid-2015 or thereabouts.

Where are we with this function? Will the KSK be rolled? How will it be done?

The KSK Design Team Report

In March 2016 ICANN published a report of a Design Team that had been considering this activity over the previous 15 months. The report is published at: https://www.iana.org/reports/2016/root-ksk-rollover-design-20160307.pdf. This report followed a public consultation in 2012, a detailed engineering study in 2013 and a study by the ICANN Security and Stability Committee in 2013 (https://www.icann.org/en/system/files/files/sac-063-en.pdf), so the activity to design the KSK roll started well before the five year anniversary, and continues today.

The first question is why all the effort with rolling the Root Zone KSK key? DNSSEC Keys, both Zone Signing Keys (ZSKs) and KSKs roll all the time, and generally they don't require the same level of care and attention stretching across a number of years. Why is this particular KSK so special?

Why is the Key Signing Key of the Root Zone of the DNS so special?

How does this particular key differ from all other keys in DNSSEC?

The simple answer is because this is the one key in the entire DNSSEC framework that has no "superior key".

In the case of a ZSK for a zone, the superior key is the KSK of the zone. A key roll of the ZSK conventionally involves introducing the new ZSK by publishing the new key in the zone's DNSKEY set (signed by the KSK, as normal). After a period to allow old cached version of the DNSKEY record to propagate to all of the zone's Authoritative Servers, plus the Time To Live (TTL) value of the key set, the new ZSK is ready for use. At this point all the signatures in the zone can be resigned using the new ZSK. The old ZSK is kept in the DNSKEY set to allow previously cached copies of signature records (signed by the old key) to be validated by the old ZSK. Again, after a propagation and TTL period the final step can be executed, which entails removal of the old ZSK from the zone. As long as there is a KSK that signs across the ZSK, rolling the ZSK is quite straightforward (Figure 1).

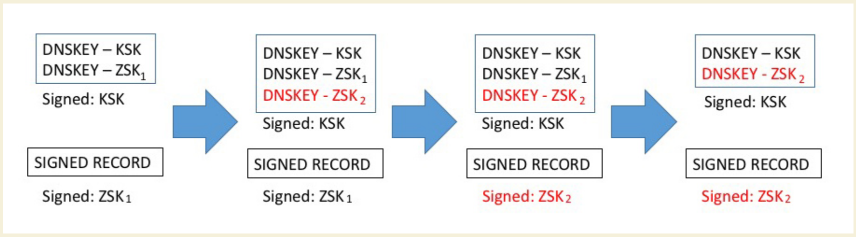

Figure 1 – ZSK Roll

Equally, rolling a KSK can be quite straightforward. The first step is to introduce the new KSK into the DNSKEY set. At this point a second RRSIG of the DNSKEY record can also be added, generated using the new KSK. While the parent zone DS record references the old KSK, the new RRSIG will not be used in validation, and nothing essentially changes. When the parent zone picks up the new DS record (corresponding to the new KSK) it can publish it immediately. The current zone has to keep publishing the old KSK and its signature value for at least the TTL of the old DS record, to serve validation correctly for any cached DS values. One the TTL of the DS has expired, the old KSK, and its corresponding signature record, can be removed from the zone (Figure 2).

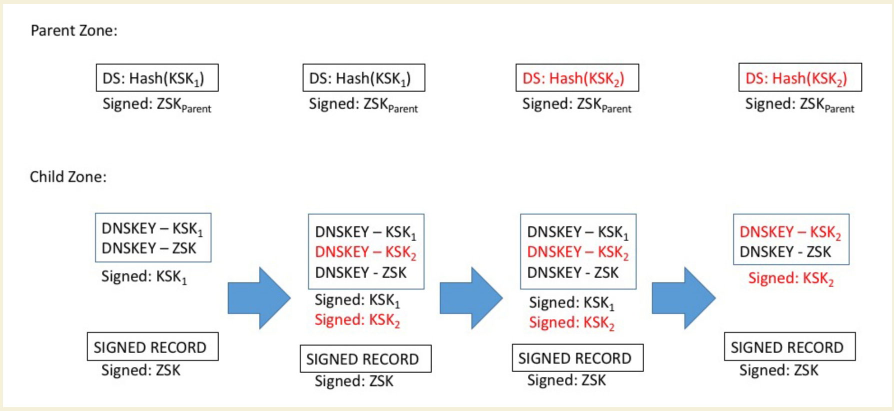

Figure 2 – KSK Roll

These steps, and various alternatives, are considered in some detail RFC6781.

The Root Zone KSK is subtly different because it has no superior key that can be used to "anchor" the rolling key. There is no superior key to sign over the KSK, and no conventional way to assure continuity of trust.

The Root Zone KSK is "special" because it has no superior key that can "anchor" the key roll. Every DNSSEC-validating resolver has a locally cached copy of this KSK value as its trusted key, and the challenge is to perform a set of changes that can signal to these resolvers that they should load a new key into its local cache as a trusted key, and do so in a secure manner.

Here is where the procedures described in RFC5011 some into play. Because there is no superior key to anchor the rolling key, RFC5011 defines a process where the old KSK is effectively the anchor point of the key roll, allow resolvers to trust the incoming KSK on the basis of their trust in the incumbent KSK. This process entails the outgoing KSK to sign across the incoming KSK, and then hold this state for an extended period (the 30 day "Add Hold-Down" period) as a means of mitigating certain forms of attack with a compromised key set.

The steps in the KSK roll process, as proposed by RFC5011, are shown in Figure 3. The changes in the Root Zone required to roll the KSK concern only changes to the DNSKEY Resource Record set (RRset) in the Root Zone. No other part of the Root Zone is altered by this key roll.

In examining the steps in Figure 3, at the outset the DNSKEY RRset contains the incumbent KSK and ZSK keys, and the entire DNSKEY record is signed by the incumbent KSK.

The first step is to introduce the new KSK into the Root Zone. This is achieved by adding an additional DKSKEY RR to the root zone, effectively publishing the new key value. The DNSKEY RRset is still signed by the incumbent KSK key. This state is held constant for no less than 30 days (the "Add Hold-Down" period defined by RFC5011), which is intended to mitigate some of the risks posed by a compromised key. Validating resolvers should validate this DNSKEY RRset, and if this is valid, and the new KSK has been stable for the Add Hold-Down period, then they are in a position to add the new KSK to their local trusted key set.

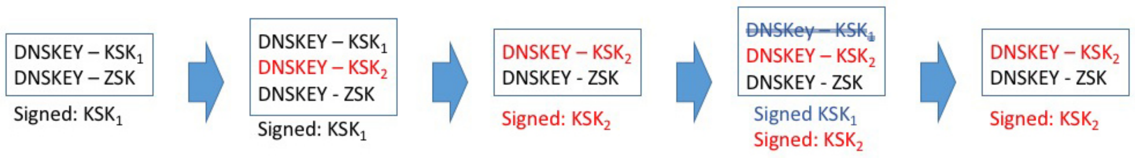

Figure 3 – Root Zone KSK Roll – after RFC5011

The second step is to remove the old KSK from the root zone. This entails removing the old KSK from the DNSKEY RRset, and signing the DNSKEY RRset with the new KSK.

One final step remains, and that is to direct resolvers to remove the old key value from their local trusted key set. This requires the old KSK to be added back into the DNSKEY RRset, this time with the REVOKE bit set, and sign the DNSKEY RRset with both the old and new KSKs. The old KSK can then be removed and destroyed.

There are a number of observations on this process.

The first is that the second and third steps can be swapped. In other words, once the new key has been introduced into the zone in Step 1, the old key can be revoked, and the DNSKEY RRset can be signed by both the outgoing and incoming key. After a suitable period (longer than the zone's Time to Live) the old KSK and its signature can be removed from the root zone. However, if one were to do this then there would be no way to back out if the process was generating an unacceptable level of user-visible failure. By splitting the withdrawal of the old key and its subsequent revocation into discrete events there is the possibility that the old (and still trusted) key can be restored of the introduction of the new KSK causes an unanticipated level of DNS failure.

The second is that there is no period where the old and new KSKs "overlap". The transition between the second and third steps involves a complete removal of the old KSK. There is no overlap at this stage where both the old and new KSKs simultaneously sign the DNSKEY RRset. The reason for this omission is that a dual-sign of the DNSKEY RRset achieves nothing in terms of assisting the roll of the KSK and could conceivably make things slightly worse. At the end of step one all resolvers will still be using the outgoing KSK to validate the DNSKEY signature. As of step two those resolvers who have added the incoming KSK to their trust set will use that incoming key, while other resolvers will be stranded without a trust point. Adding a phase of dual signing is no different to continuing the situation in step 1. Those resolvers that have not learned to trust the new KSK, and those resolvers that have picked up the new KSK as a trust point, are unable to signal their trust state to any third party, so the outside environment is none the wiser. The downside is that the dual signatures add to the size of the DNSKEY query response without aiding or even informing the key roll process.

The overall KSK roll process not quite as simple as that described above, as there is also the ZSK to consider. The ZSK is rolled every quarter. For the 10 days preceding the quarter the new ZSK is added to the DNSKEY RRset along with the incumbent ZSK, effectively priming resolvers with the new ZSK value. On the first day of each quarter the ZSK is rolled, and the root zone is signed with the new ZSK. The old ZSK remains in the DNSKEY RRset for a further 10 days to allow resolvers with cached material signed by the outgoing ZSK to still validate this cached information against the root zone DNSKEY RRset. After 10 dates the old ZSK is removed from the root zone and the ZSK roll is complete. If we use the ZSK transition days to also perform the steps of the KSK roll, then the envisaged process is more like Figure 4.

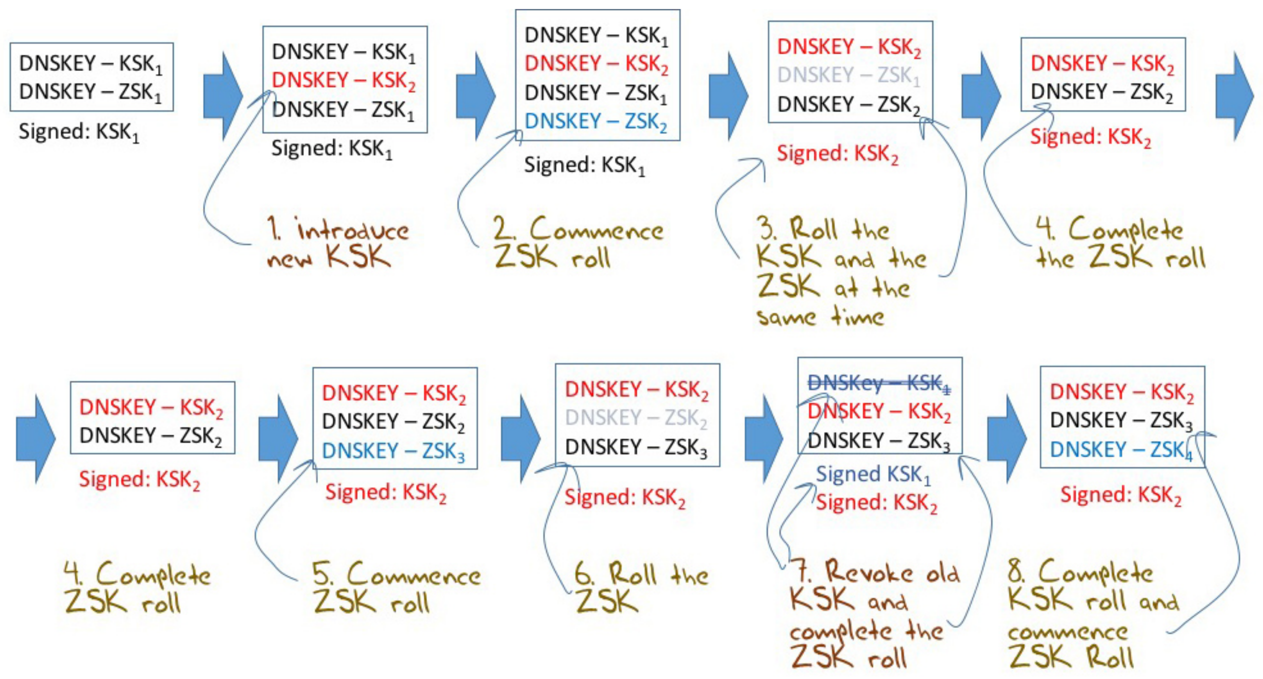

Figure 4 – Root Zone KSK and ZSK Roll

The KSK Roll Timetable

It has yet to be confirmed, but a tentative timetable for the KSK roll has been proposed in the Design Team Report (Recommendation 17 in the Design Team's report).

The recommended schedule is as follows:

1 April 2016
Prepare the new KSK key set, promulgate this to the secondary key storage facilities, and generate the root zone material to be used in the key roll steps.

11 January 2017
Introduce the new KSK value in to the Root Zone (Figure 4, Step 1).

1 April 2017
Swap out the old KSK value, and sign the Root Zone DNSKEY RRset with the new KSK value (Figure 4, Step 3).

11 July 2017

Re-publish the old KSK with the REVOKE bit set, and sign the DNSKEY RRset with both the outgoing and new KSK keys (Figure 4, Step 7).

19 September 2017

Completion: Remove the old KSK from the Root zone (Figure 4, Step 8).

The long lead time of 9 months across 2016 reflects the periodic schedule of key access ceremonies. The subsequent 9 months in 2017 is interlocked against scheduled changes of the root Zone ZSK, with KSK changes intended to slot in between ZSK changes where possible so as to ensure that the DNS response sizes do not become overly large.

What Changes and What Does Not

The KSK is an asymmetric RSA 2048-bit key. Neither the size of the key, nor the cryptographic algorithm used to generate the key, are proposed to be changed at this juncture.

A 2048-bit key is considered by a number of agencies to provide adequate key strength for the next 10 to 15 years. The Design Team report cites reports from ECRYPT, NIST and ANSSI to justify this view.[1] On this basis there is no persuasive argument to be made to justify lengthening the RSA key size.

An alternative to these large RSA Keys is to move to a different crypto algorithm. A potential alternative to RSA is one of the set of Elliptical Curve Digital Signature Algorithms (ECDSA), which offers far smaller key sizes for similar cryptographic strength. For example, an ECDSA key using curve P-256 has an equivalent strength of at least 3072-bit asymmetric RSA key. However, while this can reduce the size of DNS responses during the key roll, the downside is that the support for ECDSA in validating resolvers is by no means universal. Recent measurements point to a gap of one in every six users are using resolvers that can validate a signed response when the signature algorithm is RSA, but cannot validate an ECDSA-signed response. Changing protocols runs the risk of effectively turning off DNSSEC validation for a class of DNS resolvers and the users that lie behind these DNS resolvers.

The outcome of these considerations is that in this instance, namely the first KSK roll, the change introduced in the key roll is deliberately limited to the key value, with no change to either the protocol or the key size. This is an expression of a conservative operational principle that if you are going to embark on a change that has never been undertaken in the past, then limiting the change to just one attribute limits the risk of a negative outcome. That does not imply that the exercise is risk free. There are some considerable risks associated with this key roll.

Risks

There are a number of aspects of risk associated with this key roll.

Before going into the risks, the first step is to quantify the extent of the risk.

How many users could potentially be affected by a roll of the KSK? These days some 87% of all unique queries seen at authoritative name servers will include both the EDNS0 option and have the DNSSEC OK bit set. In other words, 9 out of 10 users channel their query to resolvers that ask authoritative name servers for the DNSSEC digital signature. But asking for the data and using the data appear to be

[1] http://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf
http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf
http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf

two different concepts. Only one third of these queries, or the DNS resolvers used by some 30% of all users, will generate a subsequent set of queries that attempt to validate the result using DNSSEC. This measurement of 30% of all users is closer to the estimate of the overall pool of risk for a KSK roll.

However, the risk of disruption is probably not quite as large as this. If the result of the validation effort is validation failure, then one half of these query streams will switch over to using a non-validating resolver, while only the remaining half, or 15% of all users, will accept the validation failure.

From this data we can observe that a change of the trust anchor material will potentially impact some 30% of the Internet user population. Given that one half of these users will switch over to a non-validating upon failure, then this implies that the worse case scenario is some 15% of all users may be impacted by this key roll to the extent that if their resolvers all lose their KSK trust point, then these users would be unable to resolve all DNS names~~ that are DNSSEC-signed~~.

~~The second part of this sentence is equally important when attempting to quantify risk. It is not the case that the worst case scenario is 15% of the Internet user population being left stranded with no DNS service at all. It means that these users will be unable to resolve DNS names that are DNSSEC-signed. Exact numbers are not available, but it does appear that while a significant proportion of DNS resolvers are willing to validate a DNSSEC-signed response, a far smaller proportion of DNS names are in fact signed. So this theoretical maximum of 15% of users who would be potentially impacted is mitigated by the observation that a far smaller proportion of DNS names are DNSSEC-signed. There are no precise public measurement measurements available on the extent of name signing and the correlation of signed names and user-driven name resolution queries, the signed name space appears to represent a very small proportion of the total DNS name space.~~

I'll leave the struck out text in place here, as an illustration that where the DNS is concerned one should always be extremely careful about making assumptions. My assumption, that this would only impact the resolution of DNSSEC-signed names is in fact not the case. Because the root zone is signed then a DNSSEC validating resolver will attempt to validate that an ostensibly unsigned DNS response is truly unsigned, and is not the result of some third party attempt to corrupt the DNS resolver. So the resolver will attempt to find the point in the name resolution path where the DNSSEC signing state switches from signed to not-signed, and validate that state, thereby assuring itself that the response is truly unsigned. This means that any of these DNSSEC validating resolvers will fail to provide any answer at all once its trust anchor state is not the same as the root. If the client turns off all DNSSEC checking (setting the Checking Disabled bit in EDNS0) then the resolver will answer all queries, but in all other cases, for both signed and unsigned names, it will return the SERVFAIL response. This increases the risk factor of failure significantly.

What could possibly go wrong?

The major risk is failure to load the new KSK as a trusted key due to failure to follow the RFC5011 procedure. One class of vulnerable DNS resolvers has DNSSEC validation enabled but does not have RFC5011 support. This may be a relatively rare combination, but the Internet supports a very broad diversity of software, and it would be rash to say that such a combination simply does not exist. It probably does exist, but quantifying how many resolvers fall into this category is far harder. Perhaps a more common occurrence would be those DNS resolvers with local configuration state that turns off "auto-managed" trusted keys. This setting effectively turns off a resolver from loading the new KSK key value when it is published in the root zone. Again its simply impossible to tell at the outset how many resolvers operate with manually managed keys, and how many users lie behind these resolvers, and whether the system administrators will be on the ball in manually loading the new KSK when it is announced in the root zone. There are those resolvers that use the current trusted KSK via a local configuration file, but are activated in the closing 30 day window prior to the KSK roll. These resolvers will not see the new KSK for the Add Hold-down time of a full 30 days, yet be expected to follow the KSK roll nevertheless. And finally there are those resolvers using an out of date configuration environment and are activated after the KSK roll. These resolvers will be stranded and will be unable to come up as a DNSSEC validating resolver until the new KSK value is loaded into their local configuration.

The second major risk is failure to load the new KSK due to problems with large DNS responses. The size of the signed response to the query for the Root Zone's DNSKEY RRset normally varies between 736 and 883 octets. The larger size occurs during the ZSK roll when 2 ZSK values are loaded into the DNSKEY RRset. There are some expectations of problems occurring with UDP when the response size starts to get to the 1,232 octet payload limit of unfragmented UDP in IPv6. There is also the common limit of 1,500 as a maximum IP packet size in much of the Internet, which corresponds to a 1,452 octet UDP payload in IPv6 and a 1,472 octet UDP payload in IPv4. When the new KSK is introduced to the DNSKEY RRset the signed response size will be 1,011 octets, and in the final 10 day period immediately prior to the KSK switch there will be 2 KSK values and 2 ZSK values, making a total of 1,158 octets of DNS payload. Revocation of the old KSK calls for the old KSK value to be added to the DNSKEY RRset, and signed by both the old and incoming KSKs. This is the largest response point, and the size of the DNSKEY query response is 1,297 octets.

Of course this assumes that the Root Zone ZSK is a constant 1024 bit value. If the ZSK increases in size, then the corresponding packet size has to increase. While the 2048 bit KSK is not seen to be a problem these days, the same cannot be said for the 1024-bit ZSK. While no one is claiming to have performed a factorization of such a key size in any useful time right now, 1024-bit keys are no longer considered to be appropriately strong by many public agencies[2]. A shift to a 2048-bit RSA key for the ZSK will add an additional 128 octets to these responses sizes.

It is possible that there are network paths that will not pass a UDP packet with a payload size of 1,297 octets. Indeed, it is also possible that there may be network paths that present issues to UDP packets with a payload size of 1,158 octets. In such a case the resolver may try dropping the EDNS0 buffer size as a hint to the authoritative name server to reduce the amount of material loaded into the additional section in an effort to reduce the response size. If at this point the server is unable to perform this reduction it will send back a truncated UDP response, and thereby signal to the resolver that it should re-try the query using a TCP transport. Here again we may anticipate to see some issues. DNS over TCP is not universally supported, and this too may fail.

Plan B

So if we can confidently anticipate some level of "damage" in this KSK roll, how much damage is too much damage and what can be done about it?

We expect that some resolvers will be unable to resolve DNS names when the KSK is switched (Step 3 in Figure 4). In this case the damage would be relatively immediate and a subset of DNSSEC-validating DNS resolvers would be unable to resolve any DNSSEC-signed name. Its likely that many of these affected resolvers could be fixed an a very short order by manually loading the new KSK key as a trusted key, or by reconfiguring them not to perform DNSSEC validation at all.

There is a second period of vulnerability where we anticipate that some DNS resolvers will be unable to resolve names when the old KSK is revoked (Step 7 of Figure 4). The reason is that during this period the DNSKEY response will grow to 1,297 octets, and this will cause some resolvers a problem, particularly in the case of UDP over IPv6, as this response exceeds the assured unfragmented IPv6 packet size of 1280 octets when the UDP and IPv6 headers (48 octets) are added to this DNS payload. Experimental studies (see section 6.1.2 of the Design Team Report) indicate that "This 1% of resolvers

[2] For example, see the NIST publication 800-57, Part, rev 4, where it notes that "algorithm/key-size combinations that have an estimated security strength of less than 112 bits [which includes RSA 2014 bit keys] are no longer approved for applying cryptographic protection on [US] Federal government information." http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

who failed consistently two or more times were used by slightly less than 3,000 end systems, or 0.04% of the sampled end system population."

As part of a potential "Plan B" in the KSK roll the design team report recommended that:

> Recommendation 14: In order to support a number of potential operational contingencies that may require rollback of changes to the root zone during each phase of the KSK key roll, SKRs [Signed Key Requests] generated using the incumbent KSK, SKRs generated using both the incumbent and the incoming KSK, and SKRs generated using the incoming KSK should be generated. The Design Team also recommends that the dual signing approach is the preferred mechanism to respond to a requirement to perform a rollback in Quarter 2 of the key roll procedure.
>
> > [This is recommending that the key material should be prepared in advance so that if it is through necessary to back out of the KSK roll, the key material that would allow this would be already available to the Root Zone administrators.]
>
> Recommendation 15: The Root Zone Management partners should undertake or commission a measurement program that is capable of measuring the impact of changes to resolvers' DNSSEC validation behavior, and also capable of estimating the population of endpoints that are negatively impacted by changes to resolvers' validation behavior.
>
> > [This is recommending that the Root Zone Managers should commission continuous measurement across the KSK roll process.]
>
> Recommendation 16: Rollback of a step in the key roll process should be initiated if the measurement program indicated that a minimum of 0.5% of the estimated Internet end user population has been negatively impacted by the change 72 hours after each change has been deployed into the root zone.
>
> > [This final recommendation is the definition of the threshold of "damage" in this context.]

Why 0.5%? The measurement techniques are relatively coarse, and its only by running a measurement over an extended period can the coarse nature of the measurement be refined to ever higher levels of granularity. If we want to measure validating DNSSEC performance on a level of days then the experimental uncertainty is +/- 0.1%, so thresholds of 0.4% or lower are unmeasurable with any reasonable level of confidence. So 0.5% is proposed as being just above the minimum measurable threshold if we are using a day-by-day measurement technique.

Why 72 hours? We anticipate that in the first day or two those operators of DNS resolvers who are impacted will take measures to correct the problem themselves and thereby restore service to their users. Any long term damage that is not being corrected by local actions will likely be evident after 72 hours.

Next Steps

This is not the end of the story by any means. There is still more work to do.

The Design Team's report needs to be carefully considered by the Root Zone Managers, and the risks associated with the key roll need to be carefully considered. It may not start on 1 April 2016 as proposed in the Design Team's report, but it will likely start in the coming months. There is little to be gained in further waiting, and it is likely that the risk factors may increase over time if the use of DNSSEC validation by resolvers and DNSSEC-signing by domain names both increase. And of course there is a certain level of expectation that has been raised by the Practice Statement. So its likely that the KSK roll will proceed.

At the same time we probably need to think about the issue of resolvers and trust points. Part of the unknown factor here is the inability of resolvers to report on their trust points. If resolvers were able to report on their trust points, presumably via some EDNS0 option embedded in queries to the root servers, then it would be possible to track the extent to which the publication of a new KSK was being picked up by resolvers as a new trusted key. Of course this would not eliminate the guesswork behind the risk assessment, as we would shift from a 2-way classification of resolvers (those who have picked up the new KSK as a trusted key vs those who have not) to a 4-way classification (multiplying the previous classification by those resolvers who are capable of reporting their trusted keys vs those who do not).

We cannot keep on increasing the RSA key size given the constraints in UDP packet size and the vagaries of UDP packet fragmentation. Maybe the next KSK should roll to ECDSA, and perhaps we should think about rolling the ZSK to ECDSA at the same time.

There is also no provision for any form of emergency KSK roll in the current environment. If the in-use KSK is no longer accessible, or it is ever compromised, then the old-signs-new model of trust transitivity cannot be used to promote a new KSK. Perhaps its time to think about pre-provisioning KSKs, and maintaining a set of KSKs that have been announced for the Add Hold-down period. This way if the current KSK is no longer accessible, or is compromised in other ways, it would be possible to switch to a pre-provisioned KSK with a very short lead time.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990's. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001 and chaired a number of IETF Working Groups. He has worked as an Internet researcher, as an ISP systems architect and a network operator at various times.

www.potaroo.net

Disclaimer