DNSOP
Internet-Draft
Intended status: Standards Track
Expires: December 13, 2018

G. Huston
J. Damas
APNIC
W. Kumari
Google

June 11, 2018

A Root Key Trust Anchor Sentinel for DNSSEC
draft-ietf-dnsop-kskroll-sentinel-14

Abstract

The DNS Security Extensions (DNSSEC) were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. These digital signatures can be verified by building a chain of trust starting from a trust anchor and proceeding down to a particular node in the DNS. This document specifies a mechanism that will allow an end user and third parties to determine the trusted key state for the root key of the resolvers that handle that user's DNS queries. Note that this method is only applicable for determining which keys are in the trust store for the root key.

[This document is being collaborated on in Github at: https://github.com/APNIC-Labs/draft-kskroll-sentinel. The most recent version of the document, open issues, etc should all be available here. The authors (gratefully) accept pull requests. RFC Editor, please remove text in square brackets before publication.]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 13, 2018.

Table of Contents

1. Introduction 2
1.1. Terminology 4
2. Sentinel Mechanism in Resolvers 4
2.1. Preconditions 5
2.2. Special Processing 5
3. Sentinel Tests for a Single DNS Resolver 6
3.1. Forwarders 8
4. Sentinel Tests for a Set of Resolvers 9
4.1. Test Scenario and Objective 9
4.2. Test Assumptions 10
4.3. Test Procedure 10
5. Security Considerations 12
6. Privacy Considerations 12
7. Implementation Experience 12
8. IANA Considerations 13
9. Acknowledgements 13
10. Change Log 14
11. References 17
11.1. Normative References 17
11.2. Informative References 18
Appendix A. Protocol Walkthrough Example 18
Authors' Addresses 21

1. Introduction

The DNS Security Extensions (DNSSEC) [RFC4033], [RFC4034] and [RFC4035] were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. DNSSEC uses Key Tags to efficiently match signatures to the keys from which they are generated. The Key Tag is a 16-bit value computed from the RDATA portion of a DNSKEY RR using a formula found in "Key

Tag Calculation" (Appendix B of "Resource Records for the DNS Security Extensions" [RFC4034]), a formula similar to a ones-complement checksum. RRSIG RRs contain a Key Tag field whose value is equal to the Key Tag of the DNSKEY RR that validates the signature.

This document specifies how security-aware DNS resolvers that perform validation of their responses can respond to certain queries in a manner that allows an agent performing the queries to deduce whether a particular key for the root has been loaded into that resolver's trusted key store. This document also describes a procedure where a collection of resolvers can be tested to determine if at least one of these resolvers has loaded a given key into its trusted key store. These tests can be used to determine whether a certain root zone KSK is ready to be used as a trusted key, within the context of a planned root zone KSK key roll.

There are two primary use cases for this mechanism:

- Users may wish to ascertain whether their DNS resolution environment resolvers is ready for an upcoming root KSK rollover.

- Researchers want to perform Internet-wide studies about the proportion of users who will be negatively impacted an upcoming root KSK rollover.

The mechanism described in this document meets both of these use cases. This new mechanism is OPTIONAL to implement and use, although for reasons of supporting broad-based measurement techniques, it is strongly preferred that configurations of DNSSEC-validating resolvers enabled this mechanism by default, allowing for local configuration directives to disable this mechanism if desired.

The KSK sentinel tests described in this document use a test comprising of a set of DNS queries to domain names that have special values for the left-most label. The test relies on recursive resolvers supporting a mechanism that recognises this special name pattern in queries, and under certain defined circumstances will return a DNS SERVFAIL response code (RCODE 2), mimicking the response code that is returned by security-aware resolvers when DNSSEC validation fails.

If a browser or operating system is configured with multiple resolvers, and those resolvers have different properties (for example, one performs DNSSEC validation and one does not), the sentinel test described in this document can still be used, but it makes a number of assumptions about DNS resolution behaviour that may not necessarily hold in all environments. If these assumptions do

not hold (such as, for example, requiring the stub resolver to query the next recursive resolver in the locally configured set upon receipt of a SERVFAIL response code) then this test may produce indeterminate or inconsistent results. In some cases where these assumptions do not hold, repeating the same test query set may generate different results.

Note that the sentinel mechanism described here measures a very different (and likely more useful) metric than [RFC8145]. RFC 8145 relies on resolvers reporting towards the root servers a list of locally cached trust anchors for the root zone. Those reports can be used to infer how many resolvers may be impacted by a KSK roll, but not what the user impact of the KSK roll will be.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Sentinel Mechanism in Resolvers

DNSSEC-Validating resolvers that implement this mechanism MUST perform validation of responses in accordance with the DNSSEC response validation specification [RFC4035].

This sentinel mechanism makes use of two special labels:

- root-key-sentinel-is-ta-<key-tag>

- root-key-sentinel-not-ta-<key-tag>

Note that the <key-tag> is specified in the DNS label as unsigned decimal integer (as described in [RFC4034], section 5.3), but zero-padded to five digits (for example, a Key Tag value of 42 would be represented in the label as 00042).

These labels trigger special processing in the validating DNS resolver when responses from authoritative servers are received. Labels containing "root-key-sentinel-is-ta-<key-tag>" is used to answer the question "Is this the Key Tag of a key which the validating DNS resolver is currently trusting as a trust anchor?" Labels containing "root-key-sentinel-not-ta-<key-tag>" is used to answer the question "Is this the Key Tag of a key which the validating DNS resolver is *not* currently trusting as a trust anchor?"

2.1. Preconditions

All of the following conditions must be met to trigger special processing inside resolver code:

- The DNS response is DNSSEC validated.
- The result of validation is "Secure".
- The Checking Disabled (CD) bit in the query is not set.
- The QTYPE is either A or AAAA (Query Type value 1 or 28).
- The OPCODE is QUERY.
- The leftmost label of the original QNAME (the name sent in the Question Section in the original query) is either "root-key-sentinel-is-ta-<key-tag>" or "root-key-sentinel-not-ta-<key-tag>".

If any one of the preconditions is not met, the resolver MUST NOT alter the DNS response based on the mechanism in this document.

2.2. Special Processing

Responses which fulfil all of the preconditions in Section 2.1 require special processing, depending on leftmost label in the QNAME.

First, the resolver determines if the numerical value of <key-tag> is equal to any of the Key Tag values of an active root zone KSK which is currently trusted by the local resolver and is stored in its store of trusted keys. An active root zone KSK is one which could currently be used for validation (that is, a key that is not in either the AddPend or Revoked state as described in [RFC5011]).

Second, the resolver alters the response being sent to the original query based on both the left-most label and the presence of a key with given Key Tag in the trust anchor store. Two labels and two possible states of the corresponding key generate four possible combinations summarized in the table:

Label	Key is trusted	Key is not trusted
is-ta	return original answer	return SERVFAIL
not-ta	return SERVFAIL	return original answer

Instruction "return SERVFAIL" means that the resolver MUST set RCODE=SERVFAIL (value 2) and the ANSWER section of the DNS response

MUST be empty, ignoring all other documents which specify content of the ANSWER section.

3. Sentinel Tests for a Single DNS Resolver

This section describes the use of the sentinel detection mechanism against a single DNS recursive resolver in order to determine whether this resolver is using a particular trust anchor to validate DNSSEC-signed responses.

Note that the test in this section applies to a single DNS resolver. The test described in Section 4 applies instead to a collection of DNS resolvers, as might be found in the DNS configuration of an end-user environment.

The critical aspect of the DNS names used in this mechanism is that they contain the specified label for either the positive and negative test as the left-most label in the query name.

The sentinel detection procedure can test a DNS resolver using three queries:

- A query name containing the left-most label "root-key-sentinel-is-ta-<key-tag>". This corresponds to a a validly-signed RRset in the zone, so that responses associated with queried names in this zone can be authenticated by a DNSSEC-validating resolver. Any validly-signed DNS zone can be used for this test.

- A query name containing the left-most label "root-key-sentinel-not-ta-<key-tag>". This is also a validly-signed name. Any validly-signed DNS zone can be used for this test.

- A query name that is signed with a DNSSEC signature that cannot be validated (described as a "bogus" RRset in Section 5 of [RFC4033], when, for example, an RRset is not signed with a valid RRSIG record).

The responses received from queries to resolve each of these names can be evaluated to infer a trust key state of the DNS resolver.

An essential assumption here is that this technique relies on security-aware (DNSSEC validating) resolvers responding with a SERVFAIL response code to queries where DNSSEC checking is requested and the response cannot be validated. Note that a slew of other issues can also cause SERVFAIL responses, and so the sentinel processing may sometimes result in incorrect or indeterminate conclusions.

To describe this process of classification, DNS resolvers are classified by five distinct behavior types using the labels: "Vnew", "Vold", "Vind", "nonV", and "other". These labels correspond to resolver system behaviour types as follows:

Vnew: A DNS resolver that is configured to implement this mechanism and has loaded the nominated key into their local trusted key stores will respond with an A or AAAA RRset response for the associated "root-key-sentinel-is-ta" queries, SERVFAIL for "root-key-sentinel-not-ta" queries and SERVFAIL for the signed name queries that return "bogus" validation status.

Vold: A DNS resolver that is configured to implement this mechanism and has not loaded the nominated key into their local trusted key stores will respond with an SERVFAIL for the associated "root-key-sentinel-is-ta" queries, an A or AAAA RRset response for "root-key-sentinel-not-ta" queries and SERVFAIL for the signed name queries that return "bogus" validation status.

Vind: A DNS resolver that has is not configured to implement this mechanism will respond with an A or AAAA RRset response for "root-key-sentinel-is-ta", an A or AAAA RRset response for "root-key-sentinel-not-ta" and SERVFAIL for the name that returns "bogus" validation status. This set of responses does not give any information about the trust anchors used by this resolver.

nonV: A non-security-aware DNS resolver will respond with an A or AAAA record response for "root-key-sentinel-is-ta", an A record response for "root-key-sentinel-not-ta" and an A or AAAA RRset response for the name that returns "bogus" validation status.

other: There is the potential to admit other combinations of responses to these three queries. While this may appear self-contradictory, there are cases where such an outcome is possible. For example, in DNS resolver farms what appears to be a single DNS resolver that responds to queries passed to a single IP address is in fact constructed as a a collection of slave resolvers, and the query is passed to one of these internal resolver engines. If these individual slave resolvers in the farm do not behave identically, then other sets of results can be expected from these three queries. In such a case, no determination about the capabilities of this DNS resolver farm can be made.

Note that SERVFAIL might be cached according to Section 7 of [RFC2308] for up to 5 minutes and a positive answer for up to its TTL.

If a client directs these three queries to a single resolver, the responses should allow the client to determine the capability of the resolver, and if it supports this sentinel mechanism, whether or not it has a particular key in its trust anchor store, as in the following table:

	Query		
	is-ta	not-ta	bogus
Type Vnew	A	SERVFAIL	SERVFAIL
Type Vold	SERVFAIL	A	SERVFAIL
Type Vind	A	A	SERVFAIL
Type nonV	A	A	A
Type other	*	*	*

Vnew: The nominated key is trusted by the resolver.

Vold: The nominated key is not yet trusted by the resolver.

Vind: There is no information about the trust anchors of the resolver.

nonV: The resolver does not perform DNSSEC validation.

other: The properties of the resolver cannot be analyzed by this protocol.

3.1. Forwarders

There is also the common case of a recursive resolver using a forwarder.

If the resolver is non-validating, and it has a single forwarder, then the resolver will presumably mirror the capabilities of the forwarder target resolver.

If the validating resolver has a forwarding configuration, and uses the CD bit on all forwarded queries, then this resolver is acting in a manner that is identical to a standalone resolver.

A more complex case is where all of the following conditions hold:

- Both the validating resolver and the forwarder target resolver support this trusted key sentinel mechanism

- The local resolver's queries do not have the CD bit set

- The trusted key state differs between the forwarding resolver and the forwarder target resolver

In such a case, either the outcome is indeterminate validating ("Vind"), or a case of mixed signals such as SERVFAIL in all three responses, ("other") which is similarly an indeterminate response with respect to the trusted key state.

4. Sentinel Tests for a Set of Resolvers

The description in Section 3 describes a trust anchor test that can be used in the simple situation where the test queries were being passed to a single recursive resolver that directly queries authoritative name servers.

However, the common end user scenario is where a user's local DNS resolution environment is configured to use a set of recursive resolvers. The single resolver test technique will not function reliably in such cases, as a a SERVFAIL response from one resolver may cause the local stub resolver to repeat the query against one of the other configured resolvers and the results may be inconclusive.

In describing a test procedure that can be used in this environment of a set of DNS resolvers there are some necessary changes to the nature of the question that this test can answer, the assumptions about the behaviour of the DNS resolution environment, and some further observations about potential variability in the test outcomes.

4.1. Test Scenario and Objective

This test is not intended to expose which trust anchors are used by any single DNS resolver.

The test scenario is explicitly restricted to that of the KSK environment where a current active KSK (called "KSK-current") is to be replaced with a new KSK (called "KSK-new"). The test is designed to be run between when KSK-new is introduced into the root zone and when the root zone is signed with KSK-new.

The objective of the test is to determine if the user will be negatively impacted by the KSK roll. A "negative impact" for the user is defined such that all the configured resolvers are security-aware resolvers that perform validation of DNSSEC-signed responses, and none of these resolvers have loaded KSK-new into their local trust anchor set. In this situation, it is anticipated that once the KSK is rolled the entire set of the user's resolvers will not be able to validate the contents of the root zone and the user is likely to

lose DNS service as a result of this inability to perform successful DNSSEC validation.

4.2. Test Assumptions

There are a number of assumptions about the DNS environment used in this test. Where these assumptions do not hold, the results of the test will be indeterminate.

- When a recursive resolver returns SERVFAIL to the user's stub resolver, the stub resolver will send the same query to the next resolver in the locally configured resolver set. It will continue to do this until it gets a non-SERVFAIL response or until it runs out of resolvers to try.

- When the user's stub resolver passes a query to a resolver in the configured resolver set, it will get a consistent answer over the timeframe of the queries. This assumption implies that if the same query is asked by the same stub resolver multiple times in succession to the same recursive resolver, the recursive resolver's response will be the same for each of these queries.

- All DNSSEC-validating resolvers have KSK-current in their local trust anchor cache.

There is no current published measurement data that indicates to what extent the first two assumptions listed here are valid, and how many end users may be impacted by these assumptions. In particular, the first assumption, that a consistent SERFAIL response will cause the local stub DNS resolution environment to query all of its configured recursive resolvers before concluding that the name cannot be resolved, is a very critical assumption for this test.

4.3. Test Procedure

The sentinel detection process tests a DNS resolution environment with three query names:

- A query name that is signed with a DNSSEC signature that cannot be validated (described as a "bogus" RRset in Section 5 of [RFC4033], when, for example, an RRset is not signed with a valid RRSIG record).

- A query name containing the left-most label "root-key-sentinel-not-ta-<key-tag-of-KSK-current>". This name MUST be a validly-signed. Any validly-signed DNS zone can be used for this test.

- A query name containing the left-most label "root-key-sentinel-is-ta-<key-tag-of-KSK-new>". This name MUST be a validly-signed. Any validly-signed DNS zone can be used for this test.

The responses received from queries to resolve each of these names can be evaluated to infer a trust key state of the user's DNS resolution environment.

The responses to these queries are described using a simplified notation. Each query will either result in a SERFVAIL response (denoted as "S"), indicating that all of the resolvers in the recursive resolver set returned the SERVFAIL response code, or result in a response with the desire RRset value (denoted as "A"). The queries are ordered by the "invalid" name, the "not-ta" label, then the "is-ta" label, and a triplet notation denotes a particular response. For example, the triplet "(S S A)" denotes a SERVFAIL response to the invalid query, a SERVFAIL response to the "not-ta" query and a RRset response to the "is-ta" query.

The set of all possible responses to these three queries are:

(A * *): If any resolver returns an "A" response for the query for the invalid name, then the resolver set contains at least one non-validating DNS resolver, and the user will not be impacted by the KSK roll.

(S A *): If any of the resolvers returns an "A" response the the "not-ta" query, then at least one of the resolvers does not recognise the sentinel mechanism, and the behaviour of the collection of resolvers during the KSK roll cannot be reliably determined.

(S S A): This case implies that all of the resolvers in the set perform DNSSEC-validation, all of the resolvers are aware of the sentinel mechanism, and at least one resolver has loaded KSK-new as a local trust anchor. The user will not be impacted by the KSK roll.

(S S S): This case implies that all of the resolvers in the set perform DNSSEC-validation, all of the resolvers are aware of the sentinel mechanism, and none of the resolvers has loaded KSK-new as a local trust anchor. The user will be negatively impacted by the KSK roll.

5. Security Considerations

This document describes a mechanism to allow users to determine the trust anchor state of root zone key signing keys in the DNS resolution system that they use. If the user executes third party code, then this information may also be available to the third party.

The mechanism does not require resolvers to set otherwise unauthenticated responses to be marked as authenticated, and does not alter the security properties of DNSSEC with respect to the interpretation of the authenticity of responses that are so marked.

The mechanism does not require any further significant processing of DNS responses, and queries of the form described in this document do not impose any additional load that could be exploited in an attack over the normal DNSSEC validation processing load.

6. Privacy Considerations

The mechanism in this document enables third parties (with either good or bad intentions) to learn something about the security configuration of recursive DNS resolvers. That is, someone who can cause an Internet user to make specific DNS queries (e.g. via web-based advertisements or javascript in web pages), can, under certain specific circumstances that includes additional knowledge of the resolvers that are invoked by the user, determine which trust anchors are configured in these resolvers. Without this additional knowledge, the third party can infer the aggregate capabilities of the user's DNS resolution environment, but cannot necessarily infer the trust configuration of any recursive name server.

7. Implementation Experience

[RFC Editor: Please remove before publication. As this section will be removed, it is more conversational than would appear in a published doc.]

List of known resolver implementations (alphabetical):

BIND Ondrej Sury of ISC reported to the DNSOP Working Group in April 2018 that this technique was peer-reviewed and merged into BIND master branch with the intent to backport the feature into older release branches. The merge request: https://gitlab.isc.org/isc-projects/bind9/merge_requests/123 Information on configuring this can be found in the BIND 9.13.0 Administrator Reference Manual (ARM), available at https://ftp.isc.org/isc/bind9/9.13.0/doc/arm/Bv9ARM.pdf

Knot resolver Petr Spacek implemented early versions of this technique into the Knot resolver, identified a number of places where it wasn't clear, and provided very helpful text to address these issues and make the document mode clear. Petr also identified an embarrassingly large number of typos (and similar) in the ksk-test setup. More information is at http://knot-resolver.readthedocs.io/en/stable/modules.html#sentinel-for-detecting-trusted-keys

Unbound Benno Overeinder of NLnet Labs reported to the DNSOP Working Group in April 2018 an intention to support this technique in Unbound in the near future. This is now implemented in Unbound version 1.7.1, available from http://unbound.nlnetlabs.nl/download.html . Configuration information is at http://unbound.nlnetlabs.nl/documentation/unbound.conf.html

A (partial) list of "client" / user side implementations (the author was keeping a more complete list of implementations, but has misplaced it - apologies, I'm happy to re-add them if you send me a note.):

http://www.ksk-test.net An Javascript implementation of the client side of this protocol is available at: http://www.ksk-test.net

http://test.kskroll.dnssec.lab.nic.cl/ Hugo Salgado-Hernandez has created an implementation at http://test.kskroll.dnssec.lab.nic.cl/

http://sentinel.research.icann.org/ The code for this implementation is published at https://github.com/paulehoffman/sentinel-testbed

8. IANA Considerations

[Note to IANA, to be removed prior to publication: there are no IANA considerations stated in this version of the document.]

9. Acknowledgements

This document has borrowed extensively from [RFC8145] for the introductory text, and the authors would like to acknowledge and thank the authors of that document both for some text excerpts and for the more general stimulation of thoughts about monitoring the progress of a roll of the KSK of the root zone of the DNS.

The authors would like to thank Joe Abley, Mehmet Akcin, Mark Andrews, Richard Barnes, Ray Bellis, Stephane Bortzmeyer, David Conrad, Ralph Dolmans, John Dickinson, Steinar Haug, Bob Harold, Wes Hardaker, Paul Hoffman, Matt Larson, Jinmei Tatuya, Edward Lewis,

George Michaelson, Benno Overeinder, Matthew Pounsett, Hugo Salgado-Hernandez, Andreas Schulze, Mukund Sivaraman, Petr Spacek, Job Snijders, Andrew Sullivan, Ondrej Sury, Paul Vixie, Duane Wessels and Paul Wouters for their helpful feedback.

The authors would like to especially call out Paul Hoffman and Duane Wessels for providing comments in the form of a pull request.

10. Change Log

RFC Editor: Please remove this section!

Note that this document is being worked on in GitHub - see Abstract. The below is mainly large changes, and is not authoritative.

From -13 to -14:

- Addressed nits from Bob Harold - https://mailarchive.ietf.org/arch/msg/dnsop/j4Serw0z24o470AnlD8ISo8o9k4

- Formatting changes (and a bit more text) in the implementation section.

- Closes PR #21: Clarify indeterminate and resolution systems,

- Closes PR #22: Updates to -13 describing the test procedure for a set of resolvers

- Closes PR #23: Fix sundry typos,

- Closes PR #24: Editorial and clarifications to the new text

- Closes PR #25: Clarified when the test can be run

From -12 to -13:

- Merged Paul Hoffmans PR#19, PR#20.

- Moved toy ksk-test.net to implementation section.

- Split the test procedures between the test of a single DNS resolvers and the test of a collection of DNS resolvers as would be found in an end user environment.

From -11 to -12:

- Moved the Walkthrough Example to the end of the document as an appendix.

- Incorporated changes as proposed by Ondrej Sury, relating to a consistent use of Key Tag and a reference to the definition of a Bogus RRset.

- Corrected minor typos.

- Revised the Privacy Considerations.

- In response to a request from DNSOP Working Group chairs, a section on reported Implementation Experience has been added, based on postings to the DNSOP Working Group mailing list.

From -10 to -11:

- Clarified the preconditions for this mechanism as per Working Group mailing list discussion.

- Corrected minor typo.

From -09 to -10:

- Clarified the precondition list to specify that the resolver had performed DNSSEC-validation by setting the AD bit in the response

- Clarified the language referring to the operation of RFC8145 signalling.

From -08 to -09:

- Incorporated Paul Hoffman's PR # 15 (Two issues from the Hackathon) - https://github.com/APNIC-Labs/draft-kskroll-sentinel/pull/15

- Clarifies that the match is on the *original* QNAME.

From -08 to -07:

- Changed title from "A Sentinel for Detecting Trusted Keys in DNSSEC" to "A Root Key Trust Anchor Sentinel for DNSSEC".

- Changed magic string from "kskroll-sentinel-" to "root-key-sentinel-" -- this time for sure, Rocky!

From -07 to -06:

- Addressed GitHub PR #14: Clarifications regarding caching and SERVFAIL responses

- Addressed GitHub PR #12, #13: Clarify situation with multiple resolvers, Fix editorial nits.

From -05 to -06:

- Paul improved my merging of Petr's text to make it more readable. Minor change, but this is just before the cut-off, so I wanted it maximally readable.

From -04 to -05:

- Incorporated Duane's #10

- Integrated Petr Spacek's Issue - https://github.com/APNIC-Labs/draft-kskroll-sentinel/issues/9 (note that commit-log incorrectly referred to Duane's PR as number 9, it is actually 10).

From -03 to -04:

- Addressed GitHub pull requests #4, #5, #6, #7 #8.

- Added Duane's privacy concerns

- Makes the use cases clearer

- Fixed some A/AAAA stuff

- Changed the example numbers

- Made it clear that names and addresses must be real

From -02 to -03:

- Integrated / published comments from Paul in GitHub PR #2 - https://github.com/APNIC-Labs/draft-kskroll-sentinel/pull/2

- Made the Key Tag be decimal, not hex (thread / consensus in https://mailarchive.ietf.org/arch/msg/dnsop/Kg7AtDhFRNw31He8n0_bMr9hBuE)

From -01 to 02:

- Removed Address Record definition.

- Clarified that many things can cause SERVFAIL.

- Made examples FQDN.
- Fixed a number of typos.
- Had accidentally said that Charlie was using a non-validating resolver in example.
- [TODO(WK): Doc says Key Tags are hex, is this really what the WG wants?]
- And active key is one that can be used *now* (not e.g AddPend)

From -00 to 01:

- Added a conversational description of how the system is intended to work.
- Clarification that this is for the root.
- Changed the label template from _is-ta-<key-tag> to kskroll-sentinel-is-ta-<key-tag>. This is because BIND (at least) will not allow records which start with an underscore to have address records (CNAMEs, yes, A/AAAA no). Some browsers / operating systems also will not fetch resources from names which start with an underscore.

11. References

11.1. Normative References

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <https://www.rfc-editor.org/info/rfc5011>.

11.2. Informative References

[RFC8145] Wessels, D., Kumari, W., and P. Hoffman, "Signaling Trust Anchor Knowledge in DNS Security Extensions (DNSSEC)", RFC 8145, DOI 10.17487/RFC8145, April 2017, <https://www.rfc-editor.org/info/rfc8145>.

Appendix A. Protocol Walkthrough Example

This Appendix provides a non-normative example of how the sentinel mechanism could be used, and what each participant does. It is provided in a conversational tone to be easier to follow. The examples here all assume that each person has just one resolver, or a system of resolvers that have the same properties.

Alice is in charge of the DNS root KSK (Key Signing Key), and would like to roll / replace the key with a new one. She publishes the new KSK, but would like to be able to predict / measure what the impact will be before removing/revoking the old key. The current KSK has a Key Tag of 11112, the new KSK has a Key Tag of 02323. Users want to verify that their resolver will not break after Alice rolls the root KSK key (that is, starts signing with just the KSK whose Key Tag is 02323).

Bob, Charlie, Dave, Ed are all users. They use the DNS recursive resolvers supplied by their ISPs. They would like to confirm that their ISPs have picked up the new KSK. Bob's ISP does not perform validation. Charlie's ISP does validate, but the resolvers have not yet been upgraded to support this mechanism. Dave and Ed's resolvers have been upgraded to support this mechanism; Dave's resolver has the new KSK, Ed's resolver hasn't managed to install the 02323 KSK in its trust store yet.

Geoff is a researcher, and would like to both provide a means for Bob, Charlie, Dave and Ed to be able to perform tests, and also would like to be able to perform Internet-wide measurements of what the impact will be (and report this back to Alice).

Geoff sets an authoritative DNS server for example.com, and also a webserver (www.example.com). He adds three address records to example.com:

```
bogus.example.com.  IN AAAA 2001:db8::1
```

root-key-sentinel-is-ta-02323.example.com. IN AAAA 2001:db8::1

root-key-sentinel-not-ta-11112.example.com. IN AAAA 2001:db8::1

Note that the use of "example.com" names and the addresses here are examples. In a real deployment, the domain names need to be under control of the researcher, and the addresses must be real, reachable addresses.

Geoff then DNSSEC signs the example.com zone, and intentionally makes the bogus.example.com record have bogus validation status (for example, by editing the signed zone and entering garbage for the signature). Geoff also configures his webserver to listen on 2001:db8::1 and serve a resource (for example, a 1x1 GIF, 1x1.gif) for all of these names. The webserver also serves a webpage (www.example.com) which contains links to these 3 resources (http://bogus.example.com/1x1.gif, http://root-key-sentinel-is-ta-02323.example.com/1x1.gif, http://root-key-sentinel-not-ta-11112.example.com/1x1.gif).

Geoff then asks Bob, Charlie, Dave and Ed to browse to www.example.com. Using the methods described in this document, the users can figure out what their fate will be when the 11112 KSK is removed.

Bob is not using a validating resolver. This means that he will be able to resolve bogus.example.com (and fetch the 1x1 GIF) - this tells him that the KSK roll does not affect him, and so he will be OK.

Charlie's resolvers are validating, but they have not been upgraded to support the KSK sentinel mechanism. Charlie will not be able to fetch the http://bogus.example.com/1x1.gif resource (the bogus.example.com record is bogus, and none of his resolvers will resolve it). He is able to fetch both of the other resources - from this he knows (see the logic in the body of this document) that he is using validating resolvers, but at least one of these resolvers is not configured to perform sentinel processing. The KSK sentinel method cannot provide him with a definitive answer to the question of whether he will be impacted by the KSK roll.

Dave's resolvers implement the sentinel method, and have picked up the new KSK. For the same reason as Charlie, he cannot fetch the "bogus" resource. His resolver resolves the root-key-sentinel-is-ta-02323.example.com name normally (it contacts the example.com authoritative servers, etc); as it supports the sentinel mechanism, just before Dave's recursive resolver sends the reply to Dave's stub, it performs the KSK Sentinel check. The QNAME starts with "root-key-

sentinel-is-ta-", and the recursive resolver does indeed have a key with the Key Tag of 02323 in its root trust store. This means that that this part of the KSK Sentinel check passes (it is true that Key Tag 02323 is in the trust anchor store), and the recursive resolver replies normally (with the answer provided by the authoritative server). Dave's recursive resolver then resolves the root-key-sentinel-not-ta-11112.example.com name. Once again, it performs the normal resolution process, but because it implements KSK Sentinel (and the QNAME starts with "root-key-sentinel-not-ta-"), just before sending the reply, it performs the KSK Sentinel check. As it has the key with key-tag 11112 in it's trust anchor store, the answer to "is this *not* a trust anchor" is false, and so the recursive resolver does not reply with the answer from the authoritative server - instead, it replies with a SERVFAIL (note that replying with SERVFAIL instead of the original answer is the only mechanism that KSK Sentinel uses). This means that Dave cannot fetch "bogus", he can fetch "root-key-sentinel-is-ta-02323", but he cannot fetch "root-key-sentinel-not-ta-11112". From this, Dave knows that he is behind an collection of resolvers that all validate, all have the key with key tag 11112 loaded and at least one of these resolvers has loaded the key with key-tag 02323 into its local trust anchor cache, Dave will not be impacted by the KSK roll.

Just like Charlie and Dave, Ed cannot fetch the "bogus" record. This tells him that his resolvers are validating. When his (sentinel-aware) resolvers performs the KSK Sentinel check for "root-key-sentinel-is-ta-02323", none of them have loaded the new key with key-tag 02323 in their local trust anchor store. This means check fails, and Ed's recursive resolver converts the (valid) answer into a SERVFAIL error response. It performs the same check for root-key-sentinel-not-ta-11112.example.com, and as all of Ed's resolvers both perform DNSSEC validation and recognise the sentinel label Ed will be unable to fetch the "root-key-sentinel-not-ta-11112" resource. This tells Ed that his resolvers have not installed the new KSK and he will be negatively implacted by the KSK roll..

Geoff would like to do a large scale test and provide the information back to Alice. He uses some mechanism such as causing users to go to a web page to cause a large number of users to attempt to resolve the three resources, and then analyzes the results of the tests to determine what percentage of users will be affected by the KSK rollover event.

This description is a simplified example - it is not anticipated that Bob, Charlie, Dave and Ed will actually look for the absence or presence of web resources; instead, the webpage that they load would likely contain JavaScript (or similar) which displays the result of the tests, sends the results to Geoff, or both. This sentinel

mechanism does not rely on the web: it can equally be used by trying to resolve the names (for example, using the common "dig" command) and checking which result in a SERVFAIL.

Authors' Addresses

Geoff Huston

Email: gih@apnic.net
URI: http://www.apnic.net

Joao Silva Damas

Email: joao@apnic.net
URI: http://www.apnic.net

Warren Kumari

Email: warren@kumari.net