DNSOP
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

G. Huston
J. Damas
APNIC
W. Kumari
Google

March 5, 2018

# A Sentinel for Detecting Trusted Keys in DNSSEC
# draft-ietf-dnsop-kskroll-sentinel-05

## 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 determing which keys are in the trust store for the root key.

There is an example / toy implementation of this at http://www.ksk-test.net .

[ 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. Text in square brackets will be removed before publication. ]

[ NOTE: This version uses the labels "kskroll-sentinel-is-ta-<key-tag>", "kskroll-sentinel-not-ta-<key-tag>"; older versions of this document used "_is-ta-<key-tag>", "_not-ta-<key-tag>". Also note that the format of the tag-index is now zero-filled decimal. Apolgies to those who have began implmenting.]

## Status of This Memo

Table of Contents

## 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 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 validating resolvers can respond to certain queries in a manner that allows a querier to deduce whether a particular key for the root has been loaded into that resolver's trusted key store. In particular, this response mechanism 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 key roll by this resolver.

There are two primary use cases for this mechanism:

- o Users want to know whether the resolvers they use are ready for an upcoming root KSK rollover

- o Researchers want to perform Internet-wide studies about the percentage of users who will be ready for 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.

### 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. Protocol Walkthrough Example

[Ed note: This is currently towards the front of the document; we will make it an appendix at publication time, but until then it is worth having up front, as it makes the rest of the document much easier to understand ]

This section 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.

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:

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

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

   kskroll-sentinel-not-ta-02323.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 much be real, reachable addresses.

Geoff then DNSSEC signs the example.com zone, and intentionally makes the invalid.example.com record invalid/bogus (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://invalid.example.com/1x1.gif, http://kskroll-sentinel-is-ta-02323.example.com/1x1.gif, http://kskroll-sentinel-not-ta-02323.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 invalid.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://invalid.example.com/1x1.gif resource (the invalid.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 below) that he is using legacy, validating resolvers. The KSK sentinel method cannot provided him with a definitive answer to the question of what root trust anchors this resolver is using.

Dave's resolvers implement the sentinel method, and have picked up the new KSK. For the same reason as Charlie, he cannot fetch the "invalid" resource. His resolver resolves the kskroll-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 server send the reply to Dave's stub, it performs the KSK Sentinel check (see below). The QNAME starts with "kskroll-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 kskroll-sentinel-not-ta-02323.example.com name. Once again, it performs the normal resolution process, but because it implements KSK Sentinel (and the QNAME starts with "kskroll-sentinel-not-ta-"), just before sending the reply, it performs the KSK Sentinel check. As it has 02323 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 "invalid", he can fetch "kskroll-sentinel-is-ta-02323", but he cannot fetch "kskroll-sentinel-not-ta-02323". From this, Dave knows that he is behind an upgraded, validating resolver, which has successfully installed the new, 02323 KSK.

Just like Charlie and Dave, Ed cannot fetch the "invalid" record. This tells him that his resolvers are validating. When his

(upgraded) resolver performs the KSK Sentinel check for "kskroll-sentinel-is-ta-02323", it does *not* have the (new, 02323) KSK in it's 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 kskroll-sentinel-not-ta-02323.example.com; as it does not have the 02323 KSK, it is true that this is not a trust anchor for it, and so it replies normally. This means that Ed cannot fetch the "invalid" resource, he also cannot fetch the "kskroll-sentinel-is-ta-02323" resource, but he can fetch the "kskroll-sentinel-not-ta-02323" resource. This tells Ed that his resolvers have not installed the new KSK.

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.

The above 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.

Note that the sentinel mechanism described here measures a very different (and likely more useful) metric than [RFC8145]. RFC 8145 relies on resolvers reporting the list of keys that they have to root servers. That reflects on how many resolvers will be impacted by a KSK roll, but not what the user impact of the KSK roll will be.

## 3. Sentinel Mechanism in Resolvers

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

This sentinel mechanism makes use of two special labels:

o kskroll-sentinel-is-ta-<key-tag>

o kskroll-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 (i.e: 42 would be represented as 00042).

These labels trigger special processing in the resolver as specified bellow. The labels containing "is-ta" and "not-ta" are used to answer questions "Is (or is not, respectivaly) this the key tag a trust anchor which the validating DNS resolver is currently trusting?" Processing of both labels has the very same preconditions for both labels and differs only in last step.

The use of the positive question and its inverse allows for queries to detect whether resolvers support this sentinel mechanism.

## 3.1. Preconditions

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

a. DNS response for particular query is DNSSEC validated and result of validation is SECURE.

b. QTYPE is A or AAAA (Query Type value 1 or 28)

c. The OPCODE is QUERY

d. The leftmost label of the QNAME is either "kskroll-sentinel-is-ta-<tag-index>" or "kskroll-sentinel-not-ta-<tag-index>"

If all preconditions are not met, the resolver MUST NOT alter the DNS response.

## 3.2. Special processing

Responses which fullfill all preconditions in section 3.1 are subject of special processing, depending on leftmost label of the QNAME.

First, the resolver determines if the numerical value of <key-tag> is equal to any of the key tags of an active Root Zone Key Signing Key which is currently trusted by the local resolver and is stored in its store of trusted keys. An active key 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]).

As second step, the resolver alters response depending on meaning of the label and presence of key with given keytag among trusted keys. Two labels and two possible states of the keytag generate four possible combinations summarized in the table:

|  | Keytag trusted |  |
|---|---|---|
| label type | yes | no |
| is-ta | return original answer | return SERVFAIL |
| not-ta | return SERVFAIL | return original answer |

## 4. Processing Sentinel Results

This proposed test that uses the sentinel detection mechanism described in this document is based on the use of three DNS names that have three distinct DNS resolution behaviours. The test is intended to allow a user or a third party to determine the state of their DNS resolution system, and, in particular, whether or not they are using validating DNS resolvers that use a particular trust anchor for the root zone.

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 process uses a test with three query names:

o A query name containing the left-most label "kskroll-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.

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

o A query name that is signed with a DNSSEC signature that cannot be validated (such as if the corresponding RRset is not signed with a valid RRSIG record).

The responses received from queries to resolve each of these names would allow us to infer a trust key state of the resolution environment. The techniques describes in this document rely on (DNSSEC validating) resolvers responding with SERVFAIL (RCODE 2) to valid answers. Note that a slew of other issues can also cause SERVFAIL responses, and so the sentinel processing may sometimes result in incorrect conclusions.

To describe this process of classification, we can classify resolvers into four distinct behavior types, for which we will use the labels: "Vnew", "Vold", "Vleg", and "nonV". These labels correspond to resolver behaviour types as follows:

Vnew: A DNSSEC-Validating resolver that is configured to implement this mechanism has loaded the nominated key into its local trusted key store will respond with an A or AAAA RRset response for "kskroll-sentinel-is-ta" queries, SERVFAIL for "kskroll-sentinel-not-ta" queries and SERVFAIL for the invalidly signed name queries.

Vold: A DNSSEC-Validating resolver that is configured to implement this mechanism that has not loaded the nominated key into its local trusted key store will respond with an SERVFAIL for "kskroll-sentinel-is-ta" queries, an A or AAAA RRset response for "kskroll-sentinel-not-ta" queries and SERVFAIL for the invalidly signed name queries.

Vleg: A DNSSEC-Validating resolver that does not implement this mechanism will respond with an A or AAAA RRset response for "kskroll-sentinel-is-ta", an A or AAAA RRset response for "kskroll-sentinel-not-ta" and SERVFAIL for the invalid name.

nonV: A non-DNSSEC-Validating resolver will respond with an A or AAAA record response for "kskroll-sentinel-is-ta", an A record response for "kskroll-sentinel-not-ta" and an A or AAAA RRset response for the invalid name.

Given the clear delineation amongst these three cases, if a client directs these three queries to a simple resolver, the variation in response to the three queries should allow the client to determine the category of the resolver, and if it supports this mechanism, whether or not it has a particular key in its trust anchor store.

```
                                Query
                   +----------+-----------+------------+
                   |  is-ta   |  not-ta   |  invalid   |
           +-------+----------+-----------+------------+
           | Vnew  |    A     |  SERVFAIL |  SERVFAIL  |
           | Vold  | SERVFAIL |      A    |  SERVFAIL  |
      Type | Vleg  |    A     |      A    |  SERVFAIL  |
           | nonV  |    A     |      A    |     A      |
           +-------+----------+-----------+------------+
```

A "Vnew" type says that the nominated key is trusted by the resolver and has been loaded into its local trusted key stash.  A "Vold" type says that the nominated key is not yet trusted by the resolver in its own right.  A "Vleg" type does not give any information about the trust anchors, and a "nonV" type indicates that the resolver does not perform DNSSEC validation.

## 5. Sentinel Test Result Considerations

The description in the previous section describes a simple situation where the test queries were being passed to a single recursive resolver that directly queried authoritative name servers, including the root servers.

There is also the common case where the end client is configured to use multiple resolvers.  In these cases the SERVFAIL responses from one resolver will prompt the end client to repeat the query against one of the other configured resolvers.

If any of the client's resolvers are non-validating resolvers, the tests will result in the client reporting that it has a non-validating DNS environment ("nonV"), which is effectively the case.

If all of the client resolvers are DNSSEC-validating resolvers, but some do not support this trusted key mechanism, then the result will be indeterminate with respect to trusted key status ("Vleg"). Simlarly, if all the client's resolvers support this mechanism, but some have loaded the key into the trusted key stash and some have not, then the result is indeterminate ("Vleg").

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 clause, then the resolver will presumably mirror the capabilities of the forwarder target resolver.  If this non-validating resolver it has multiple forwarders, then the above considerations will apply.

If the validating resolver has a forwarding configuration, and uses the CD flag on all forwarded queries, then this resolver is acting in a manner that is identical to a standalone resolver.  The same consideration applies if any one one of the forwarder targets is a non-validating resolver.  Similarly, if all the forwarder targets do not apply this trusted key mechanism, the same considerations apply.

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

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

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

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

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

## 6. Security Considerations

This document describes a mechanism to allow users and third parties to determine the trust state of root zone key signing keys in the DNS resolution system that they use.

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 the normal DNSSEC validation processing load.

## 7. Privacy Considerations

The mechansim in this document enables third parties (with either good or bad intentions) to learn something about the security configuration of recursive name servers.  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 then determine which trust anchors are configured in the user's resolver.

## 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, Andreas

Schulze, Mukund Sivaraman, Petr Spacek, Andrew Sullivan, 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. Petr Specek implmented 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 this.

## 10. Change Log

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

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 keytag 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 keytags 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

[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>.

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <https://www.rfc-editor.org/info/rfc6840>.

## 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>.

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