Domain Name System Operations
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2010

W. Wijngaards
O. Kolkman
NLnet Labs

June 29, 2010

DNSSEC Trust Anchor History Service
draft-ietf-dnsop-dnssec-trust-history-02

Abstract

When DNS validators have trusted keys, but have been offline for a longer period, key rollover will fail and they are stuck with stale trust anchors. History service allows validators to query for older DNSKEY RRsets and pick up the rollover trail where they left off.

Requirements Language

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 [RFC2119].

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 http://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 31, 2010.

Copyright Notice

1. Introduction

This memo defines a trust history service for DNS validators -- the component in a resolver that performs DNSSEC [RFC4034] validation, validator for short.

A validator that has been offline or missed an (emergency) rollover can use this service to reconfigure themselves with the current trust-anchor. Using a newly defined resource record (RR) that links old DNSKEYS together, the TALINK RR, a validator fetches old DNSKEY RRsets and checks they form a chain to the latest key (see Section 3). The lists of old DNSKEYS, linked with the TALINK RRs, do not necessarily need to be published in the zone for which the DNSKEY history is being maintained but can be published in any DNS domain. We will call the entity that offers the trust history the History Provider. The choice of the History Provider is made by the maintainer of the validator, possibly based on a hint provided, using the TALINK, by the zone owner.

Section 2 provides background on the mechanism and usage. It looks at the viewpoints of publishers and consumers of trust anchors, the use of keys with revocation flags, and SEP flags.

The algorithm that the validator uses to construct a history and reconfigure a new key is detailed in Section 4, it uses the TALINK RR type defined in Section 3. The algorithms for how providers of trust history can fetch the DNSKEY data as published by the zone they track and publish that are explained in Section 5.

2. Motivation and Description

Validators provide a service in DNSSEC that can be seen from two ways. Seen from the publisher's point of view, they provide assurance that the data as received is as it was when it left the publisher's hands. In this way of looking at things, validators provide a publication integrity service. The publisher can be confident that nobody can alter the published data (if it is

validated), because any alteration will be detected. So it protects a publisher from being seen to send someone to the wrong place.

From the consumer's point of view, validators provide a reason to trust the data from the network. In this view, the validator is

making a claim about whether the data ought to be accepted or not. This is subtly different from the publisher's point of view, because the question for the consumer is not whether the data is safe while the consumer is not looking, but whether the data is safe for the consumer at the moment of consumption. Validation protects a consumer from going to the wrong place.

These two slightly different ways of looking at the situation result in slightly different operational goals. Whereas publishers want to make assertions about their data, by controlling the roll over of keys, consumers want to get the best assurance that they can get that the data they are consuming is correct.

If a validator has been offline during a key rollover event for one of its trust anchors, then the validator will be unable to validate answers that need that trust anchor. For the publisher, this state of affairs is acceptable: the publisher is confident that no validator ever consumes the wrong data. For the consumer, however, this state of affairs represents an outage.

Since publishers of trust anchors already use a chained series of keys to perform rollovers under some circumstances (see [RFC5011]), it is possible to use the history of that chain to allow a validator to resume service for the consumer without needing to use an out-of-band mechanism to obtain a new trust anchor. This improves the experience for consumers of validated data, and increases the chances that DNSSEC is useful for consumers of DNS data.

The mechanism to do this is a double-linked list that recounts a portion of the history of DNSKEY Resource Records. The list is used by a validator to catch up with the changes that the validator somehow missed. This approach may be thought of as replaying the [RFC5011] rollover history, only at a later time.

2.1. Considerations for Using a Revoked Key

The keys that the publisher rolled are marked REVOKED by the RFC5011 protocol. At this point the publisher considers the keys revoked, but the validators have not yet seen this or marked the keys as revoked. In the RFC5011 protocol, the validators probe regularly and can then see if keys are revoked. If unable to probe, they will be unable to see if keys are revoked. Hence when using a history to recount rollovers, the consumer's validator has also missed a number of revocations. The goal is to pick up the right keys and also the new revocations along the way.

Although the keys have been marked by the publisher as REVOKED a long time ago, for the consumer these REVOKED keys are new information.

Their storage in the history list makes it possible for consumers to pick up key revocations if they missed the revocation announcement because they could not probe.

This is the allowed usage of REVOKED keys. The publisher is announcing their presence. And the validators mark them as REVOKED after verification. The initial part of this verification is the reverse walk through the history list, which is to avoid exposing which key is trusted. This means that older signatures with keys that have in the meantime been revoked are used to construct and verify the history list by the validator.

A consequence is that once a publisher marks keys as REVOKED, there will still be consumers who are using such keys, because they have not seen the revocation. From the publishers point of view they are revoked and the revocation is filed in the historical key list. From the consumers point of view, it has not seen a revocation yet, and a historical key list lookup algorithm is a state change where a new trusted key is obtained while the old key is observed to be revoked.

2.2. Motivation for Requiring the SEP Bit

The SEP bit is used to differentiate Key Signing Keys from other keys. It is defined in [RFC3757], it is used to designate trust anchors in [RFC5011]. The protocol herein specified requires that DNSKEYs that are subject to use for the trust history service have the SEP bit set. The reason for this is to keep the set of keys that need to be stored in history small.

3. The TALINK Resource Record

The DNS Resource Record type TALINK (decimal 58) ties the elements of a linked list of DNSKEY RRs together.

The rdata consists of two domain names. The first name is the start, or previous name, and the other name the end or next name in the list. The empty label '.' is used at the endpoints of the list.

The presentation format is the two domain names. The wire encoding is the two domain names, with no compression so the type can be treated according to [RFC3597]. The list is a double linked list, because this empowers low memory hosts to perform consistency checks.

The TALINK used at the zone apex holds the endpoints of the list. The TALINKs that form the lists hold previous and next entries. These TALINKs are distinguished by their usage (entrypoint or list connection). The double linked list is not circular, because lookups must stop when they reach the oldest entry.

4. Trust History Lookup

This is the algorithm that a validator uses to detect and resolve the situation in which a trust-anchor is out of sync with the DNSKEYs published by a zone owner. The algorithm uses the TALINK RR type which is used to link various old DNSKEYs as published by the History Provider, to arrive from the outdated configured Trust Anchor to one that matches the current DNSKEY. The TALINK RR type is defined in Section 3.

When the algorithm below results in failure the trust history cannot be built and a new trust anchor will need to be re-configured using another mechanism.

Step 1: The validator performs a DNSKEY lookup to the target zone, which looks like any other initial DNSKEY lookup that the validator needs to match a trust anchor to the currently used DNSKEY RR set. If the keyset verifies with the trust anchor currently held, the trust-anchor is not out of sync. Otherwise, store the DNSKEY RR set as result. The algorithm will successfully build a linked list to this DNSKEY RR, or delete the trust point, or fail.

All nameservers (the ones authoritative for the zone or the upstream resolver caches when the validator is not full resolver) SHOULD be checked to make sure the DNSKEY RR sets are the same. The results can differ if a key-rollover is in progress and not all nameservers are in sync yet. This case can be detected by checking that the older keyset signs the newer and take the newer as result keyset. If both of the keysets sign each other, the result keyset has the newest rrsig that validates it using the other keyset. Use the the average over the middle of the inception and expiration dates of the signatures that are validated (and for serial arithmetic assume all dates on these signatures lie within 2^(SERIAL_BITS-1) distance). If the keysets do not sign each other then this is not a secure change in the keyset and the history lookup fails.

Step 2: Fetch the trust history list end points. Perform a query of QTYPE TALINK and QNAME the domain name that is configured to be the History Provider for the particular domain you are trying to update the trust-anchor for.

Step 3: Go backwards through the trust history list as provided by the TALINK linked list. Verify that the keyset validly signs the next keyset. This is [RFC4034] validation, but the RRSIG expiration date is ignored. Replace the owner domain name with the target zone name for verification. One of the keys that signs

the next keyset MUST have the SEP bit set. The middle of inception and expiration date from the valid signature MUST be older than that of the signature that validates the next keys in the list. Take the average if multiple signatures validate (and for serial arithmetic assume all dates on these signatures lie within 2^(SERIAL_BITS-1) distance). Query type TALINK to get previous and next locations.

If all SEP keys have the REVOKE flag set at this step, and the keyset is signed by all SEP keys, then continue but store that the end result is that the trust point is deleted, see Section 5 [RFC5011].

If all SEP keys are of an unknown algorithm at this step, continue and at the next step, when you verify if the keyset signs validly:

if false, continue with result a failure, if true, continue with the end result that the trust point is deleted. Thus, the keysets with unknown algorithms are stepped over with an end result of failure because the validator cannot determine if unknown algorithm signatures are valid, until the oldest keyset with unknown algorithms is signed by a known algorithm and the result is set to deletion and step 3 continues to a known key.

Step 4: When the trust anchor currently held by the validator verifies the keyset, the algorithm is done. The validator SHOULD store the result on stable storage. Use the new trust anchor for validation (if using [RFC5011], put it in state VALID).

5. Trust History Tracker

External trackers can poll the target zone DNSKEY RRset regularly. Ignore date changes in the RRSIG. Ignore changes to keys with no SEP flag. Copy the newly polled DNSKEY RRset and RRSIGs, change the owner name to a new name at the history location. Publish the new RRset and TALINK records to make it the last element in the list. Update the TALINK that advertises the first and last name.

Integrated into the rollover, the keys are stored in the history and the TALINK is updated when a new key is used in the rollover process. This gives the TALINK and new historical key time to propagate.

The signer can support trust history. Trust history key sets need only contain SEP keys. To use older signers, move historical RRSIGs to another file. Sign the zone, including the TALINK and DNSKEY records. Append the historical RRSIGs to the result. Signing the zone like this obviates the need for changes to signer and server software.

6. Example

In this example the trust history for the 'example.net' zone is published in the 'example.com' namespace. The DNSKEY rdata and RRSIG rdata is omitted for brevity, it is a copy and paste of the data from example.net.

```
$ORIGIN example.com.
```

```
example.com. TALINK h0.example.com. h2.example.com.

h0 TALINK . h1.example.com.
h0 DNSKEY ...
h0 RRSIG ...

h1 TALINK h0.example.com. h2.example.com.
h1 DNSKEY ...
h1 RRSIG ...

h2 TALINK h1.example.com. .
h2 DNSKEY ...
h2 RRSIG ...
```

The example.net zone can advertise the example.com History Provider by providing the TALINK shown here at example.com at the apex of the example.net zone. The TALINK at example.com is then not needed.

7. Deployment

The trust history is advertised with TALINK RRs at the zone apex. These represent alternative history sources, that can be searched in turn. The TALINK at the zone apex contains the first and last name of the list of historical keys.

The historical list of keys grows perpetually. Since most validators have recent keys, their processing time remains similar as the list grows. If validators no longer have trust in the keys then they need no longer be published. The oldest key entries can be omitted from the list to shorten it.

The validator decides how long it trusts a key. A recommendation from the zone owner can be configured for keys of that zone, or recommendations per algorithm and key size can be used (e.g. see [NIST800-57]). If a key is older than that, trust history lookup fails with it and the trust point can be considered deleted. This assumes the validator has decided on a security policy and also can take actions when the update of the trust anchor fails. Without such policy, or if the alternative is no DNSSEC, the approach below can be used.

In general, the decision can be that any key - no matter how old or

how small - is better than no security. The validator then never considers a key too old, and the lookup algorithm becomes an unsecured update mechanism at the time where the key can be trivially broken. The history provider SHOULD provide these broken keys to facilitate clients performing the unsecured update. If a key can not be trivially broken then it provides a non-trivial amount of security that the history lookup algorithm uses to get the current keys. Conceivably after the update the result is stored on stable storage and the client is thereafter safe - it performs a leap of faith. The validator operator can opt for this set up after considering the trade-off between loss of DNSSEC, loss of connectivity, and the argument that perceived security is worse than no security.

The history lookup can be used on its own. Then, the trust history is used whenever the key rolls over and no polling is performed. The results of trust history lookup SHOULD be stored on stable storage, so that the trust history lookup does not need to be performed if the last results are okay and for use as trusted anchor in the next history lookup.

If a validator is also using [RFC5011] for the target zone, then the trust history algorithm SHOULD only be invoked if the [RFC5011] algorithm failed due to the inability to perform probes. This is the case when the last [RFC5011] successful probe was more than 30 days ago. If a new key has been announced, invoke the history if no 2 probes succeeded during the add hold-down time and there was no successful probe after the add hold-down time passed. Therefore the time of the last successful probe MUST be stored on stable storage.

For testing the potentially very infrequently used lookup, the following SHOULD be implemented. For the test the lookup is triggered manually by allowing the system to be given a particular keyset with a last successful lookup date in the past and a test History Provider. The test History Provider provides access to a generated back-dated test history.

8. Security Considerations

The History Provider only provides copies of old data. If that historic data is altered or withheld the lookup algorithm fails because of validation errors in Step 3 of the algorithm. If the History provider or a Man in the Middle Adversary (MIMA) has access to the original private keys (through theft, cryptanalisis, or otherwise), history can be altered without failure of the algorithm. Below we only consider MIMAs and assume the History Provider is a trusted party.

Spoofing by a MIMA of data looked up in step 2 or 3, i.e. spoofing of TALINK and DNSKEY data, can present some alternate history. However the DNSKEY RR set trusted that the history should arrive at is already fixed by step 1. If an attempt is made to subvert the algorithm at step 2 or 3, then the result keyset can not be replaced by another keyset unnoticed.

To change the keyset trusted as the outcome, the step 1 data has to be spoofed and the key held by the validator (or a newer historic key) has to be compromised. Unless such spoof is targeted to a specific victim, a spoof of the step 1 result has a high visibility. Since most of the validators that receive the spoof have an up-to-date trust anchor most validators that would receive this spoof return validation failure for data from the zone that contains the DNSKEYs. An adversary will therefore have to target the attack to validators that are in the process of an update. Since validators do not announce that they use trust history lookup until step 2 adversaries will not be able to select the validators.

A spoof of data in steps 2 and 3, together with a compromised (old) key, can result in a downgrade. At steps 2 and 3 a faked trust point deletion or algorithm rollover can be inserted in a fake history. This avoids the high visibility of spoofing the current key (see previous paragraph) and downgrades to insecure.

Finally there is the case that one of the keys published by the History Providers has been compromised. Since someone spoofing at step 1 of the lookup algorithm and presenting some fake history to a compromised key, of course does not include key revocations and does extend the history to contain the compromised key, it therefore is not really useful for a History Provider to remove the key from the published history. That only makes lookups fail for those validators who are not under attack. Useful action could be to update validators using some other means.

Rollover with [RFC5011] revokes keys after use. If a History Provider is used, then such revoked keys SHOULD be used to perform history tracking and history lookup. The trust anchor keys that the validator has in its own storage and final current keys that it stores MUST NOT be trusted if they are revoked.

If the validator operator chooses to operate trust history without also using [RFC5011] the trust anchor does not get hold-down timer protection. This has associated risks, in that the immediate rollover without timeout that it provides could be abused (if private keys are compromised). Such abuse could result in the stored lookup

results to become compromised. The key changes can be logged, to inform operators and keep an audit trail.

The SEP bit is checked to make sure that control over the KSK is necessary to change the keyset for the target zone.

The algorithm can be used to get the inception and expiration times of signatures on the current keyset, a clock. A MIMA can attempt to shorten history and put back that clock, but the algorithm attempts to make this difficult to target and highly visible to others.

If the clock of the validator can be influenced, then setting it forward is unlikely to give advantage, but setting it backward enables a replay attack of old DNSSEC data and signatures. This vulnerability exists also in plain DNSSEC.

9. IANA Considerations

Resource record type TALINK has been defined using RFC5395 expert review, it has RR type number 58 (decimal).

10. Acknowledgments

Thanks to the people who provided review and suggestions, Peter Koch, Andrew Sullivan, Joe Abley, George Barwood, Edward Lewis, Michael StJohns, Bert Hubert, Mark Andrews, Ted Lemon, Steve Crocker, Bill Manning, Eric Osterweil, Wolfgang Nagele, Alfred Hoenes, Olafur Gudmundsson, Roy Arends and Matthijs Mekking.

11. References

11.1. Informative References

[NIST800-57] Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendations for Key Management", NIST SP 800-57, March 2007.

[RFC3757] Kolkman, O., Schlyter, J., and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag", RFC 3757, April 2004.

[RFC5011] StJohns, M., "Automated Updates of DNS Security

(DNSSEC) Trust Anchors", RFC 5011, September 2007.

11.2. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

Authors' Addresses

Wouter Wijngaards
NLnet Labs
Science Park 140
Amsterdam 1098 XG
The Netherlands

EMail: wouter@nlnetlabs.nl

Olaf Kolkman
NLnet Labs
Science Park 140
Amsterdam 1098 XG
The Netherlands

EMail: olaf@nlnetlabs.nl