Domain Name System Security WG
INTERNET DRAFT
<draft-lewis-dnssig-authorization-00.txt>

Edward Lewis
Olafur Gudmundsson
Trusted Information Systems
November 21, 1997

DNSSEC Signature and Data Verification Semantics
<draft-lewis-dnssig-authorization-00.txt>

0.0 Status of this Memo

1.0 Abstract

This draft discusses authorization models for DNSSEC that can be used to determine the relationship of a KEY RR and a DNS RRset in the validation process. Is this key trusted to sign for this data? Is this data trusted because it was signed by this key? This draft defines a number of different policies that can be used and what the signing authority of keys are in each. This draft also addresses what steps are recommended in the secure DNS resolution process and how the authorization policy is put to use. The ideas and definitions expressed here are based on the authors experience in implementing a reference secure resolver.

2.0 Introduction

This draft discusses the DNSSEC [RFC2065, RFC2137] authorization model which is introduced in [SECEXT2], section 6.3.1. The draft builds upon that discussion, adding in lessons learned in implementing a resolver which performs the DNSSEC validation procedure itself.

NOTE: This draft is being issued to stimulate discussion; it is not a final word on the subject at hand. We feel that [SECEXT2] should progress without reference to the policies under which

signatures are evaluated. The specification should specify the process but not the policy. The policy should be left to a separate document.

In this draft there are two main sections. The first defines four authorization models, each defining which keys are allowed, or trusted, to sign a set of data. (Or, conversely, which sets of data can be signed by a given key.) The second main section covers the resolution algorithm implemented in the reference resolver.

The implementation serves as a model for a resolver that would be installed in the name server - which is where the DNSSEC extensions are designed to be. A secure enhancement to the resolv code will more likely be a result of transaction signatures or TSIG [TSIG].

3.0 Motivation

An RRset is received as a result of a query to a DNS name server. The set of data is accompanied by a signature. The cryptographic operation involving the signature, its generating key, and the data may or may not "be successful" - but what does that mean? Does a successful cryptographic verification mean the data is valid? Does a failure indicate the data is not valid?

By itself, the result of the cryptographic operation is not conclusive, although it is a big hint towards authentication of the data's source. It is possible for an attacker to supply a maliciously inserted malformed signature that guarantees a set will fail verification, even though the data is valid. Likewise, an attacker may supply a set of data of its own choosing, sign it with a key and supply the correctly computed signature. This results in a successful verification but invalid data.

The weakness that such an attack is exploiting is not a problem of the authentication. The problem is in the authorization. If the cryptographic operation is the only measure of authentication, then the authorization policy is "anyone can speak for anything." This is not a wise policy.

Authentication of data is fairly straightforward as described in

the DNSSEC RFCs and drafts. The authorization policy is not straightforward, and we anticipate that in the future, the policy may be site, or even resolver, specific.

4.0 The Role of the Authorization Policy

The authorization policy can be used to weed out cryptographic operations which would not result in the trusting of data. A resolver can decide not to examine a signature if the signer is not trusted for the data.

In an exchange of data, there are three elements involved that can employ an authorization model. The sender may want to limit who can speak for the validity of the data. The receiver may decide whom it trusts. In addition, the mechanism relaying and caching the data can decide whether or not to forward data based upon its authenticity.

DNSSEC documents have been focused on the receiver's policy. The sender's policy choices are limited by the DNSSEC design - there is no mechanism for a sender to prevent another from signing the data.

The policy described in section 6.3.1. is a policy fit for an "end" receiver. During the implementation of a reference resolver, we felt that the policy in 6.3.1. is not appropriate. The policy implemented is the policy labeled "DOWN" in the next section. Section 6.3.1. describes "UP" with "DOMAIN" and "TRUSTED."

The reference resolver, however, does not prevent the use of policies other than DOWN. The reference resolver has two main functions, one that performs the resolution according to the DOWN policy with an option to disable even that, and another function that will run the cryptographic operation over a given data set and a given key without checking any authorization policy.

5.0 Motivation for Multiple Signatures & Authorization Models

There are a number of valid and plausible situations in which multiple signatures and authorization models are desired.

One such situation occurs when there is a mix of signed and unsigned zones in the Internet DNS tree. A particular domain, whether representative of a single user, host, or even a whole zone, may desire to have records signed, but the zone above is not configured to sign data. In this situation one of two

configurations are possible.

One possibility is the use of an signing authority. Although the DNS medium may consider the data to be part of an unsigned zone, and thus decide to ignore any signatures attached to the data, the receiver of the data may decide to trust the signing authority and attempts to use a public key of the signing authority to verify the data cryptographically. (The DNS medium may elect to not trust a signature which is not verified according to the DNS hierarchy. This will be argued elsewhere in the document.)

Another possibility is that the entity generating the data to be placed in DNS wants to have a non-DNS signature placed on the data. The DNS signature, applied by the zone, merely claims that what was entered into DNS at one zone is what is received

by the final resolver. It does not cover the authenticity of the originating data, it trusts that the data was entered into the originating zone master file was done so faithfully. If the sender and receiver of the data "sent" via DNS is not willing to trust the DNS administrators, they could apply other signatures which are entered into the originating zone, ignored in the transit, and returned with data that has passed the other DNS authentication tests.

Yet another situation is proposed by the UP policy, introduced in the next section. It calls for a zone key to be signed by the zone above and the zone(s) below. The object of this is to avoid having to have the root keys pre-configured in every computer. According to this policy, there would be multiple signatures attached to at least the zone keys in nearly every situation.

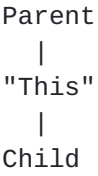
6.0 Four Policies

The four policies described here are those considered or discussed during the implementation of the reference resolver. This is not an exhaustive list of policies, nor are they the only policies envisioned to be desired by end resolvers. These, however, are the only ones specifically addressed in the draft.

To describe the policies, a few terms are defined in the context of this draft.

For the Charts below:

Parent - the zone above (the current topic zone)
Child - a zone below (the current topic zone)



Names:
Apex - a name at the top of the zone; it has the SOA record
Deleg - any name with NS records, other than the Apex,
within a zone
Other - in the name context, a domain name w/o an NS record

Types:
SOA,NS,KEY - sets of that type
NXTu - (Upper) NXT with NS bit on and SOA bit off
NXTl - (Lower) NXT with both NS and SOA bits on
NXT - NXT set which has both the NS and SOA bits off
Other - in the type context, all other types not described

Keys:
Zone key (ZK) - has zone key flags turned on and either is
signed by another zone key or is preconfigured as trusted
Signing key (SK) - has signing ability in flags and is signed
by a zone key or is preconfigured as trusted

For the rest of the discussion:
Other keys - keys held in DNS but are not marked as being
used by DNSSEC for authentication; e.g., E-mail keys.
Domain - a zone or zone member(s) and all its children zones
and (recursively) all their descendants
Approved key - a zone key or signing key that is trusted -
by virtue of being signed and validated by an authorized
zone key which is approved (in the same manner) or has
been pre-configured (hence its value is trusted)

6.1 DOWN

The DOWN policy states that the authorized signer of a record is given by the following table:

Data \ Signer	Zone's ZK	Parent's ZK	Child's ZK	Wildcard SK	Owner SK
Apex(SOA)	Y	N	N	x	N

Apex(NS)	Y	N	N	x	N
Apex(KEY)	N	Y	N	x	N
Apex(NXTu)	N	Y	N	x	N
Apex(NXTl)	Y	N	N	x	N
Apex(other)	Y	N	N	x	Y
Deleg(NS)	N	N	Y(*)	x	N
Deleg(KEY)	Y	N	N	x	N
Deleg(NXTu)	Y	N	N	x	N
Deleg(NXTl)	N	N	Y(*)	x	N
Deleg(other)	N	N	Y(*)	x	Y(*)
Other(KEY)	Y	N	x	N	N
Other(NXT)	Y	N	x	N	N
Other(other)	Y	N	x	Y	Y

* = Signature appears initially only in the child zone.
x = This key cannot exist. If, through a misconfiguration, it does, it is not authorized to sign.

To use this table, take a RRset and a key. Find the row of the table closest to the RRset (the owner name is either an apex, delegation, or "other" in the eyes of a particular zone). Then decide on the relationship of the key to the set - if a zone key, is it from the current zone, the parent, a child, or some other?

Note that names which are "Deleg" in one zone are "Apex" in another. Remember to adjust what is "zone," "parent," and "child" accordingly.

Summarizing the DOWN policy - a record set can only be signed by the zone responsible for it, or by a key matching the owner name by equality or by wildcard. There are restrictions on the SOA, NS, KEY, and NXT sets beyond this.

6.2 UP

The UP policy varies in a small way from the DOWN policy. The zone key of a zone is also signed by each child zone and the set with signature is initially available at the child.

The only line that differs from the previous chart is:

Data	Zone's ZK	Parent's ZK	Signer Child's ZK	Wildcard SK	Owner SK
...					
Apex(KEY)	N	Y	Y(+)	x	N

...

+ = multiply signed, once by each of the children zone.

One could imagine the keys for COM. being signed by the root key and then by each child zone. This sounds like more of a headache than it is. Asking for a set of COM.'s keys from a particular server could return the KEY set, including a SIG(KEY) from root and one other SIG(KEY) signed by the zone served by the name server. (Multiple, perhaps, if the server is serving multiple delegations of COM.)

Unlike the problem of RRs in a set needing to be complete, the complete set of SIG RRs is not necessary, just so that there is one that is trusted by the resolver.

6.3 DOMAIN

The DOMAIN policy has been previously called the TRUE policy. Hopefully the new name is more mnemonic.

The DOMAIN policy states that a zone key for a name is authorized to sign for all records in the domain, not just the zone. In other words, a parent's zone key can sign for records in its child zones.

The chart changes to:

Data	Zone's ZK	Parent's ZK	Signer Child's ZK	Wildcard SK	Owner SK
Apex(SOA)	Y	Y	N	x	N
Apex(NS)	Y	Y	N	x	N
Apex(KEY)	N	Y	UP/DOWN	x	N
Apex(NXTu)	N	Y	N	x	N
Apex(NXTl)	Y	Y	N	x	N
Apex(other)	Y	Y	N	x	Y
Deleg(NS)	Y	Y	Y(*)	x	N
Deleg(KEY)	Y	Y	N	x	N
Deleg(NXTu)	Y	Y	N	x	N
Deleg(NXTl)	Y	Y	Y(*)	x	N

Deleg(other)	Y	Y	Y(*)	x	Y(*)
Other(KEY)	Y	Y	N	N	N
Other(NXT)	Y	Y	N	N	N
Other(other)	Y	Y	N	Y	Y

6.4 TRUSTED

The TRUSTED policy states that certain pre-configured keys are authorized to sign for any data, any where in the tree. This policy can be used in addition to or as a replacement for UP or DOWN.

6.5 Policy summary

Note that of these four policies, only UP and DOWN are mutually exclusive. UP or DOWN maybe combined with TRUSTED and/or DOMAIN, or the latter two may be used in any combination without either UP or DOWN.

7.0 DNSSEC Secure Resolution

A secure resolver was implemented mostly to gain experience in resolving DNS signatures. The code itself is not production-quality nor ported; instead the lessons learned in the implementation are to be incorporated into the resolution in the name server code, the documentation of the specifications, and eventually the resolver libraries.

7.1 DNSSEC Authorization Policy

As DNSSEC builds upon the existing DNS model, we (the authors of the draft and the implementation) believe the authorization model used by DNSSEC should mimic the current model DNS uses to apply credibility to the data. This is interpreted as adhering to the zone boundaries in the authorization model.

Because of this, we feel that the DOWN policy is the best suited, although, for efficiency reasons, the UP model is even a better fit - even though this allows a child to sign the upper zone's keys. (Assuming we can handle unsecured parents in the

searching.)

The TRUSTED and DOMAIN policies are counter to this. They erase the zone boundaries in establishing trust. This is undesirable for debugging and liability reasons.

If there is a problem in the resolution of DNS data, the problem is more easily tracked if the verification chain follows the

already established chain of zone delegations. At each delegation there is contact information - indicating who is able to address a problem. In the event that a zone is misbehaving in a manner which warrants its discontinuance, the parent zone is the place to address this.

If zone boundaries are not maintained in the authorization model, debugging a situation could be more complex as the non-DNS relationships of the participants is revealed.

As a final caveat to this section, these comments apply just to the authorization model used with in DNS. End resolvers are expected in some situations to request unchecked data from DNS so that they can apply their own trust model.

7.2 Implementation Asides

The implementation made heavy use of caching information while resolving a query. KEY, NS, and A records were held in case they were needed in the steps described below. The implementation was successful in being able to avoid issuing a query for previously seen data, unless the held data was incorrect, incomplete, or not of sufficient credibility. In doing this, the resolution process kept network traffic to a minimum to help its performance.

If the secure resolver is merged with a name server, the need for the resolver cache is fortunately eliminated. This saves a fair amount of the implementation effort. The caches are replaced by the RR's stored in the normal course of operation of the name server.

7.3 Pre-cryptographic Sanity Checks

The response from a name server should be handled as a list of RRsets, in many cases the list has just one element. However, for queries of types "any" and "sig" and queries that include CNAME references, there will be multiple RRsets to verify.

The checks needed on each RRset are:

Does the set help answer the question?
The set must either be a match for the name, type, and class in the query, or a CNAME or negative acknowledgment (nack), such as an SOA or NXT.

If the answer is a "nack," is the message correct?
If the message is an SOA, does it correspond to the

correct zone? If an NXT, does the range from the owner to the next name cover this set, and/or is there a conflict in the bit map of types? (At a delegation point, the particular set determines which NXT should be returned.)

Are there locally trusted keys to resolve towards?
If there are none, the resolution process cannot succeed, and this kind of error/misconfiguration should be raised to the user so they can confirm the situation.

Did the answer intentionally or erroneously contain only SIGs?
We may have requested SIG data, or a name server error may have passed along the wrong records.

Should the set be signed or unsigned?
This question is further discussed in the next section.

7.4 Security Expectation Discovery

Determining whether or not a set is supposed to be signed (in the eyes of the DNSSEC transfer mechanism) begins with discovering the authoritative zone and the security status of that zone.

Discovering the zone of an RRset requires that a known zone, with known security expectations, is pre-configured and available as a starting point for the search. This search is most easily performed assuming the DOWN policy, and this is why the reference implementation decided on using the DOWN policy.

Using the DOWN policy, the starting point must be "above" the set in the DNS tree. If no point is available, then the search is abandoned. If there is a choice of starting points, the chosen point should be the closest to the set.

To best describe the search, imagine that the set is owned by e.d.c.b.a., and the starting point is .b.a. With two exceptions, the search will examine the status of c.b.a., d.c.b.a., and e.d.c.b.a. The exceptions occur when the set we are examining are the KEY or the upper NXT of a delegation point.

At each tested name, we seek a KEY record and/or an NS record. If we are in a secured zone, there will be a non-null zone key at a delegation point to a secured child. In an unsecured zone, we seek NS records to find delegations. The following chart gives the complete transition table from domain to domain:

To:	From: Secure	Unsecure	Experimental
Secure	Zone Keys	impossible	Zone Keys
Unsecure	Null Zone Keys	NS record	Null Zone Key or No Key but NS
Experi-mental	Null + Non-Null Zone Keys	impossible	Both Null and Non-Null Zone Keys
Subdomain	No zone keys, no NS	No NS	No zone keys, no NS

"Impossible" refers to the impossibility of the transition in the search given that the starting point is the closest secured zone. Once the search transitions from secured zones to unsecured zones, it stays unsecured. If zone keys are encountered, they are ignored by the search, perhaps the administrator of the configuration should contact this zone to obtain the trusted public keys out of band of DNS.

The result of this search is the security status of the set's name. The name could be a zone (being an apex) or a non-delegated subdomain. More importantly, the search can determine whether the set is expected to be signed or unsigned (zone keys are found, although they may be null), or if there is no definite answer (the set is either in an experimental zone or a zone with no [verifiable] keys).

Note: to extend this to the UP policy, there must be a way to discover the security of the parent of the preconfigured zone. If the parent (or a grand parent) is unsecured, then there is a potential for problems. More thought is required in this area before reaching a final conclusion. It has been demonstrated that the DOWN policy is feasible.

Results: A set is either SIGNED, MAYBE, UNSIGNED

7.5 Signature Evaluation

The evaluation of the signatures is independent of the security expectation. This was a surprising observation that resulted from the implementation.

Assuming multiple signatures are possible for a set of data, the

status of the collection of signatures may fall into one of four categories:

Absent - there are no signatures
Immaterial - the only signatures present were generated by keys neither approved nor authorized to sign
Validated - at least one approved and authorized key's signature validated
Failed - all approved and authorized key signatures failed the cryptographic operation

One issue sidestepped by the implementation is the case in which multiple signatures lead to a validated answer. This has an impact on the setting of the TTL and perhaps other areas.

Results: A signature set is either Absent, Failed, Immaterial, or Validated

7.6 Mapping of Expectation and Evaluation to Result

Using the last sentence of each of the previous two sections, refer to this table:

Status\ Expectation	SIGNED	MAYBE	UNSIGNED
Validated	Verified	Verified	--------
Failed	Failed	Failed	--------
Immaterial	Missing Sig're	Unsig'd Unsure	Unsig'd Sure
Absent	Missing Sig're	Unsig'd Unsure	Unsig'd Sure

The dashed results refer to situations that cannot occur in this process. If a set has been deemed to be unsigned, this means that no approved and authorized key has been found. Therefore, during the signature evaluation, nothing can be set to Validated or Failed.

The returns of Failed and Missing Signature indicated authentication failures. The other codes are successes in validating the authentication. However, this does not map one-to- one with successful and unsuccessful queries.

7.7 Repeating Attempts

If any set in the list of answers fails verification, all of the sets should be held in distrust. For example, if the first CNAME record in a chain does not validate but all others do, this could be an indication of someone trying to divert answer from the true source to an invalid source. After performing all of the cryptographic tests, one more pass is recommended to guard against this situation.

A successful authentication only indicates a successful query if at least one set in the answer matches the original criteria. That is, if a query for an address record results in a validated

address record, possibly in addition to a sequence of CNAMEs, then the query is successful. If, on the other hand, the answer set includes a successfully authenticated negative acknowledgment, the query is a failure.

With this in mind, there are a set of circumstances under which the resolver can try the query again at a different name server. The different name server should be one (or all) of the authoritative name servers for the desired set. This list of servers may have been obtained as part of the security expectation search.

For most validation failures, it is wise to retry the query. If a local misconfiguration occurs, such as not having local keys, then a retry won't help - in fact, in this case, the local resolver may choose to use the answer as is, meaning the local resolver is acting like a pre-DNSSEC resolver.

8.0 Security Considerations

This draft covers issues relating to the interpretation of the authenticity of DNS responses based upon a series of DNS queries and pre-configured data.

9.0 References

[RFC1034] RFC 1034, "Domain Names - Concepts and Facilities", P. Mockapetris, 1987, <http://ds.internic.net/rfc/rfc1034.txt>

[RFC1035] RFC 1035, "Domain Names - Implementation and Specification", P. Mockapetris, 1987, <http://ds.internic.net/rfc/rfc1035.txt>

[RFC2065] RFC 2065, "Domain Name System Security Extensions", D. Eastlake and C. Kaufman, 1997, <http://ds.internic.net/rfc/rfc2065.txt>

[RFC2137] RFC 2137, "Secure Domain Name System Dynamic Update", D. Eastlake, 1997, <http://ds.internic.net/rfc/rfc2137.txt>

[RFC2181] RFC 2181, "Clarifications to the DNS Specification", R. Elz, 1997, <http://ds.internic.net/rfc/rfc2181.txt>

[SECEXT2] "Domain Name System Security Extensions", D. Eastlake, 1997, <ftp://ietf.org/internet-drafts/draft-ietf-dnssec-secext2-01.txt>

[TSIG] "Transaction Signature", P. Vixie, O. Gudmundsson, D. Eastlake, 1997, <ftp://ietf.org/internet-drafts/draft-ietf-dnsind-tsig-0?.txt>

10.0 Author's Addresses

Edward Lewis
Trusted Information Systems
3060 Washington Road
Glenwood, MD 21738

Olafur Gudmundsson
Trusted Information Systems
3060 Washington Road
Glenwood, MD 21738

+1 301 854 5794
<lewis@tis.com>

+1 301 854 5700
<ogud@tis.com>