DNS Operations
Internet-Draft
Expires: August 17, 2005

S. Krishnaswamy
SPARTA Inc.

February 13, 2005

Split-View DNSSEC Operational Practices
draft-krishnaswamy-dnsop-dnssec-split-view-00.txt

Status of this Memo

Copyright Notice

Abstract

The security extensions to the Domain Name System (DNSSEC) allow for integrity protection, whereby it is possible to make a determination of the verity of data returned from the Domain Name System in response to a query. Current operation of the Domain Name System also allows for the creation of multiple views of data, where the answer returned in response to a query is dependent on the origin of the query. Data integrity and the ability to return possibly conflicting values as in split-views may be construed to be mutually conflicting goals; but this apparent dichotomy is resolvable in practice through proper configuration. This document provides recommendations for correctly configuring the split-view DNSSEC environment in a typical enterprise network.

Table of Contents

1. Introduction

Split-view DNS is the term used to describe multiple views of DNS information for a domain based on where and by whom the query is sent. Split-views help contain DNS names to only those portions of the network that need to see these names. Although primarily meant to be a network management technique, the tailoring of the DNS to create an internal view of information hidden from the outside is also seen by some as improving their organization's security posture, by preventing the exposure of internal host names, knowledge of whose existence is deemed to be sensitive.

Relying solely on split-view DNS to protect sensitive hosts from attacks has proven to be less than adequate in the past. Attack vectors in recent Internet exploits have been able to successfully infect hosts with or without their IP addresses being published in the DNS. Conversely, publishing the IP addresses of hosts that are otherwise secured does not necessarily increase their vulnerability to these attacks. Name hiding through split-view DNS is primarily useful as part of a more comprehensive defense-in-depth strategy to provide one line of defense against name-based attacks.

The security extensions to DNS [1] provide for origin authenticity and data integrity. These properties are determined by validating the chain-of-trust from the signed record to some trusted key configured at the end resolver. In the case of split-view DNS every chains-of-trust in every view must validate properly. Some names may be common between multiple views but contain different data. Cache pollution is a possibility when data from the wrong view is returned in response to a query. Building a chain-of-trust from a trusted key above the zone that has split views, to data in the internal view of a zone can be especially problematic, caching problems notwithstanding.

The objective of this document is to describe approaches for configuring split-view DNSSEC environments with the additional requirement that no server be both authoritative and recursive at the same time. Separation of authoritative and recursive name servers not only provides simple role separation, but is also an important security measure in DNS for protecting authoritative name servers against compromised caches.

In cases where the different views of DNS information correspond to different physical networks, the name servers authoritative for the internal and external views of data are often separated by a firewall. Among some of the frequently observed DNS resolution misbehaviour [3] is the problem of resolvers aggressively retransmitting queries from behind misconfigured firewalls that allow

queries out, but drop all returned responses. This problem is exacerbated by a handful of errant queries that are sent by only a subset of internal resolvers, which makes problem isolation extremely difficult. This document provides recommendations for reducing the impact of errant queries in the split-view DNS setup and also makes recommedations for DNS-related packet filtering rules required to support the proper operation of the suggested configuration.

Section 2 describes the general approach for configuring split-view DNS, which by itself, is independent of DNSSEC. Considerations for DNSSEC appear in Section 3 .

2. Split-view DNS

2.1 Background

Different views of the DNS can be created by a process of "query channeling". Here, different servers are made authoritative for the different views of the DNS information and queries are channeled to these name servers based upon their origination address.

It is also possible to use a single machine as the authoritative name server for both views of data by running multiple instances of the name server process on a machine with multiple network interfaces, and answering differently based on the query source. Some name server implementations also directly support split-view DNS. Variants include the view-based approach and the data tagging approach. In the former, the name server loads multiple zone databases and makes available answers from a particular zone based on the origin of the query. The second approach tags the data in the database itself as either being internally, externally or globally available.

The single name server approaches are susceptible to leakage of DNS information if the host on which they operate is compromised. Confidentiality of the namespace is directly tied to how resilient the name server is against such attacks. An alternate way to protect a namespace of sensitive hosts is to have that entire namespace reside within a private delegation. By doing so, it is possible to have the protection given to the name server that serves these names commensurate with the protection given to the hosts themselves. Since hosts in the private branch are explicitly marked as such by virtue of their domain name, this method also allows the network administrator to better classify hosts as being public or private and lessens the opportunity for sensitive hosts to be inadvertantly placed in public domains. Private delegations are useful when name hiding is the only reason for namespace separation. They have the drawback that they do not allow for transparency during name

resolution; queries have to be made for specific names in specific views.

This document describes a generic configuration for split-view DNS using multiple nameservers without relying on any special capablities from any machine or name server implementation. The architecture is modeled around a typical enterprise structure: the two views are for the internal and external portions of the network, with the external portion actually residing within the boundary network. The two networks are separated by a packet filtering firewall. A packet filtering firewall also separates the boundary network from the external Internet. Name hiding is not an objective of this split-view setup, but avoiding cache pollution is. Although the two concepts are related, this configuration is not recommended for hiding sensitive names because of the ease with which names can be leaked out due to trivial configuration errors. Again, if name hiding is the main objective for providing split-views, the approach of using a private delegation for sensitive names is strongly encouraged.

The suggested configuration uses a combination of multiple name servers and query forwarding. One name server answers queries for the internal view and forwards all requests for external data to a second name server. The second name server recursively answers queries but only if asked by the first. Other name servers are configured in such a way so as to decouple the roles of the authoritative and recursive name servers.

2.2 Query Channeling

The main DNS concern for split-views is that of preventing cache pollution. Cache pollution can be avoided by carefully controlling how the queries are sent to different name servers.

Resolving outer data is straightforward since queries follow their normative paths. For the internal view, a two level recursive server scheme is recommended. One server functions as a recursive forwarder and is responsible for answering all internal queries. This server forwards all queries for internal data to their respective authoritative name servers while recursively obtaining answers from the outside from a second-level name server. The second-level name server is a simple caching name server that asks questions from the outside, but only if asked by the recursive forwarder. The recursive forwarder and the name servers authoritative for the internal data reside in the internal network; the second-level recursive name server that is used for returning answers from the outside resides in the boundary network.

The two-level recursive scheme controls where queries are directed to. Since queries for internal data are sent to authoritative name servers which are not also recursive, this scheme also controls where data is received from. In this way internal data is kept totally separate from external data, thus preventing cache pollution. Figure 1 illustrates the above setup.

```
       ^                    root, TLD servers etc
       |                           ^ (queries)       EXTERNAL NAMESERVERS
    [Outside]                      |                    ^ (responses)
       |                           |                    |
       v                           |                    |
   ------------[O u t e r----P a c k e t---F i l t e r]-------------------
       ^                           |                    |
       |                           |                    v (queries)
       |                           |            AUTHORITATIVE EXT-VIEW SERV
       |                           v (responses)
     [Boundary]            RECURSIVE NAMESERVER
       Net                       ^ (queries from the
       |               CLIENTS   | recursive forwarder
       |       (responses ^      | protected by TSIG)
       |     for internal |      |
       v            data) |      |
   ------------[I n n e r----P a c k e t---F i l t e r]-------------------
       ^                  |      |
       |         (queries)|      |
       |                  v      v (external data)
       |            RECURSIVE FORWARDER <---------> INTERNAL
   [Internal]            ^ (internal data)   (query/   CLIENT
       |                 |                  response)
       |                 |
       |                 v
       |         AUTHORITATIVE INT-VIEW SERV
       v
```

Figure 1

It is useful to note that the internal recursive forwarder must not attempt to recursively answer queries if the authoritative name server for internal-view data fails to respond. If it did so, external data could be returned in such circumstances and lead to cache pollution. Since neither the server authoritative for a forwarded zone nor the server doing the forwarding can recursively answer queries for delegations from that zone, the internal recursive forwarder must explicitly forward queries for every internal delegation to its respective authoritative name server. This rule can be relaxed while forwarding queries to name servers that are simultaneously authoritative for the child as well as the parent

zone.

2.3 Controlling Errant Queries

DNS queries that are sent from the internal recursive forwarder to the outside should only be directed towards the second-level recursive name server. Since the second-level name server has no knowledge of internal-view data, internal resolvers must not use it directly for resolving queries. Only properly configured internal recursive forwarders must be approved to send queries to this name server and solely for the purpose of resolving external answers. It is conceivable that there are multiple approved name servers sending queries to the second-level name server and TSIG is the recommeded method for controlling which name servers can send these queries. Having these rules alternatively configured in the packet filter is also possible, but using TSIG for performing this authorization eases packet filter administration for DNS.

2.4 Name Server Requirements

This section summarizes the list of requirements for the various name servers involved in the split-view configuration.

2.4.1 Internal Recursive Forwarder

- o Ability to forward queries to specific name servers.
- o Ability to control forwarding behaviour such that the recursive option is not tried, even if the name server that queries are normally forwarded to fails to respond.
- o Ability to recursively answer queries.
- o Ability to protect the integrity of messages using TSIG for selected destinations.

2.4.2 Second-level Recursive Name Server

- o Ability to recursively answer queries.
- o Ability to verify TSIG protection on messages.
- o Ability to filter incoming queries based on the TSIG key used to protect the message.

2.4.3 Authoritative Internal and External-view Name Servers

- o Ability to authoritatively answer queries for a zone.
- o Ability to disable all recursive behaviour.

3. Split-view DNSSEC

The DNSSEC concern for split-view is ensuring that the internal and

external chains-of-trust validate properly. This concern is addressed by making an appropriate choice of trusted and SEP keys. Some DNSSEC configurations may also make the split-view setup more resilient against cache pollution.

DNSSEC forces one to think about the threat environment for DNS data in split-views. Any view that is likely to be spoofed has to be signed. Often two views of a split zone are administered separately, so having different zone signing keys for the different views is also desirable. While validating external data is relatively straightforward, there are multiple approaches that can used for validating internal data. The method of choice depends on what the threat environment for the internal view is perceived to be, the amount of end-resolver configuration overhead that is needed, the ease of debugging and the ability to have administrative separation between the two split-views. The configuration overhead at end resolvers is mainly associated with the task of defining trust anchors at different recursive resolvers. Having fewer keys is desirable in that it makes key management easier. It is also desirable to reduce the amount of reconfiguration required for clients that move between the two views of data, while still being able to tie an answer to a particular view. The different options for internal data validation are further outlined below.

3.1 No internal validation

```
          (No trusted key)           parent zone (trusted)
                                          ^
                                          |
                                          |
             split zone               split zone
              (internal)               (external)
                                          ^
                                          |
                                          |
             sub split zone          sub split zone
              (internal)               (external)
```

Figure 2

This is an option if the security requirements for the internal zone are more relaxed than the external zone. The threat environment for the internal zone in this scenario does not include DNS compromise and validation results returned from the internal recursive forwarder is not important. The internal recursive forwarder does not have any trusted key configured and does not perform any validation.

3.2 Same Key Signing

```
                        -----------> parent zone (trusted)
                       /       ^          ^
                      /        |          |
                     /       (same key)-->|
                    /                     |
             split zone               split zone
              (internal)               (external)
                  ^                       ^
                  |                       |
                  |                       |
             sub split zone          sub split zone
              (internal)               (external)
```

Figure 3

In this scenario, a single private key is used to sign both the internal and the external zone data. The glue DS and NS records at the delegation point of the split zone all correspond to the external view data. Validation proceeds by constructing two separate segments of the chain-of-trust. In the first segment, data at the level of the split and below is validated by constructing a chain-of-trust contained entirely within the internal view. If the trusted key is configured at or below the level of the split, validation stops at this point and queries are never sent to the outer view. If not, a second validation chain segment is constructed from the DS record covering the split to the trusted key. In forming the second validation segment all queries (including the query for the DS record of the split zone) are sent to the outer zone. Since the same private key is used to sign both views of data, the DS record, even though pointing to the key in the outer view of the split zone applies to the key in the internal view also, thus completing the chain-of-trust.

This approach allows flexibility in choosing the level at which the trusted key is configured, with the possibility of the same trusted key being used for validating answers on both views.

Although easy to setup, this approach is difficult to troubleshoot. There is no easy way to identify if the record obtained for a query corresponds to the internal view or the external view. Using the same key also makes administrative separation of the two views of data difficult.

3.3 **Partial Decoupling of chains-of-trust**

```
                                     parent zone


   (trusted) split zone               split zone (trusted)
              (internal)                (external)
                  ^                        ^
                  |                        |
                  |                        |
             sub split zone           sub split zone
              (internal)                (external)

                                Figure 4
```

With the DS record in the parent always pointing to a key in the outer view, the construction of the chain-of-trust becomes problematic when the keys used to sign data in the two views of the split are different. The trusted key cannot be defined above the level of the split since there would be no way of linking the DS record in the outer zone to the apex DNSKEY set in the internal view of the split zone.

A simple solution is to configure the trusted key at the level of the split such that the chains-of-trust for the internal and external zones share no common records that might cause any ambiguity.

Having separate keys for the two views of data is useful for troubleshooting and in determining which view a given record belongs to. Cache pollution can be detected because such cases would lead to validation failures.

This configuration however involves more configuration overhead since trusted keys need to be configured for every zone that is split. This problem is more pronounced when dealing with validating stub resolvers on mobile nodes, where moving between the internal and external views would involve constant reconfiguration of all of these trusted keys.

3.4 Complete Decoupling of chains-of-trust

```
   (trusted) parent zone             parent zone (trusted)
              (internal)               (external)
                  ^                        ^
                  |                        |
                  |                        |
             split zone                 split zone
              (internal)                (external)
                  ^                        ^
                  |                        |
                  |                        |
             sub split zone           sub split zone
              (internal)                (external)
```

Figure 5

One problem with Section 3.3 is that trusted keys need to be configured for every zone that is split under the parent. An option to circumvent this while still retaining the advantages of the earlier setup is to split the parent also, and configure the trusted key at the level of the parent. An internal name server is configured as the authoritative server for the internal view of this split and the internal recursive forwarder is modified to forward all internal queries for the parent zone to it.

Although this option reduces the number of trusted keys at the end resolver, the trusted key still needs to change when moving between the two views. Since splitting the parent essentially creates two new zones, records in the parent that were previously common in both views would now need to be duplicated in the two split zones. The number of such records is typically not very large, but the overhead and complexity in maintaining duplicate records can still be a burden.

3.5 Multiple DS Records

```
                        -----------> parent zone (trusted)
                       /    (DS1)         ^
                      /                   |
                     /                    | (DS2)
                    /                     |
             split zone               split zone
              (internal)               (external)
                  ^                       ^
                  |                       |
                  |                       |
             sub split zone          sub split zone
              (internal)               (external)
```

Figure 6

In this approach, the parent is not split; however, DS records corresponding to each of the two different views are published at the delegation point. The glue and NS records at the delegation point still corresponds to the external zone but this information is never used by the internal zone. As in option 4, the trusted key is configured at the level of the parent zone. The chain-of-trust from this trusted key to either zone is formed by using one of the two DS records, which ever is applicable at that view.

Since some coordination between the split zone and the parent is required to publish multiple DS records, this approach is most suitable when the split is made at a level lower than the zone apex (e.g. for example.com, the split is made at a level lower than example.com). This approach lends itself to using different keys in different views while still allowing for minimal configuration at the end resolvers; trusted keys need not be changed even if the nodes are mobile across the two views. This approach has the advantage that administrative separation of the two views of the split can be maintained while still having a single key configured at the end resolvers. Identifying which view a given record belongs to can be done by tracing back the keys used to form the chain-of-trust.

While most of the internal zone contents can be kept private to the internal view, the DS record must still be exposed. This should not be a problem since data hiding is not be the objective of the split-view setup as was mentioned before. An attendent problem with multiple DS records is that since the validation algorithm iteratively looks for a DS record in the parent while completing the chain-of-trust there is some added computational overhead which increases as the number of DS records in the delegation point grows. It may also be difficult for the parent to include all DS records

that the child sends especially when the two entities are located in different organizations. Lastly, the internal view is still susceptible to an insider "attack", where data from the outside view injected in response to internal queries can corrupt the cache. This attack is common to all scenarios that use a common key for validating internal and external zone contents. Any cache pollution introduced due to administrator errors can also escape detection for the same reason.

3.6 Name Server Requirements

All name servers listed below must conform to the specifications given in [2]. Additionally, the Internal Recursive Forwarder must support the following:

- o Ability to validate DNSSEC responses.
- o Support for configurable DNSSEC trusted keys. It should be possible to configure more than one trusted key.

4. Packet Filtering Considerations

The following subsections define the rules that must be configured in the two packet filters depicted in Figure 1 in order to support the split-view configuration.

4.1 Inner packet filter

In order to allow the above configuration to work, any packet filtering system between the internal network and the boundary network must allow all of the following types of packets.

DNS queries from any internal server to the second-level recursive name server (Finer level access control is done by TSIG):

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	internal	>1023	rec.srv	53	N/A
UDP	internal	53	rec.srv	53	N/A
TCP	internal	>1023	rec.srv	53	Any

Other queries originating from internal servers but not destined to the second-level recursive name server must be denied.

Responses to the above queries from the second-level recursive name server to any internal server:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	rec.srv	53	internal	>1023	N/A
UDP	rec.srv	53	internal	53	N/A
TCP	rec.srv	53	internal	>1023	Set

Queries from clients in the boundary network to any internal name server:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	client	>1023	internal	53	N/A
TCP	client	>1023	internal	53	Any

Responses to the above queries from the (any) recursive forwarder to clients in the boundary network:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	internal	53	client	>1023	N/A
TCP	internal	53	client	>1023	Set

4.2 Outer packet filter

Any packet filtering system configured between the boundary network and the external network needs to allow the following.

Queries from the recursive name server in the boundary network to the outside network:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	rec.srv	>1023	outside	53	N/A
UDP	rec.srv	53	outside	53	N/A
TCP	rec.srv	>1023	outside	53	Any

Responses to the above queries from the outside to the boundary network recursive name server:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	outside	53	rec.srv	>1023	N/A
UDP	outside	53	rec.srv	53	N/A
TCP	outside	53	rec.srv	>1023	Set

Queries from outside clients to the external-view authoritative servers:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	outside	>1023	auth.serv(ext view)	53	N/A
UDP	outside	53	auth.serv(ext view)	53	N/A
TCP	outside	>1023	auth.serv(ext view)	53	Any

Responses to the above queries from the external-view authoritative server to the outside:

Proto	SrcIP	SrcPort	DestIP	DstPort	AckBit
UDP	auth.serv(ext view)	53	outside	>1023	N/A
UDP	auth.serv(ext view)	53	outside	53	N/A
TCP	auth.serv(ext view)	53	outside	>1023	Set

Note that in this configuration, queries from all recursive name servers in the boundary network for any external view information would need to transit outward through the second-level packet filter and then back again into the boundary network. If the existing packet filter policy prevents such traffic patterns, all such recursive name servers would need additional forwarding statements to forward these queries directly to their respective authoritative name servers without going through the packet filter.

5. Summary

This document describes an approach for configuring split-view DNSSEC. The approach uses a two level recursive scheme where an internal recursive forwarder resolves inside answers and marshalls all outside queries to a second-level recursive name server. TSIG between the internal and the second-level name servers protects against errant queries.

The recommended configuration has been shown to be adjustable for various needs and security consideration levels. Differences in these approaches make trade-offs between configuration overhead and validation overhead. Trading-off in favour of minimal operator overhead is useful for overall maintainability of the system, especially when split-view DNS is considered in the context of nodes that are mobile across the two views.

Although split-view DNSSEC is possible using the recommended setup, it still involves significant effort: for configuring the various name servers, for setting up zone forwarding, for configuring and distributing shared keys for TSIG, and, depending on the configuration, for performing DS (or keyset) exchanges for every view of a split zone. Some configurations may also require multiple trusted keys in end resolvers which may change between views. Proper care must be taken to ensure that correct split-view behavior is consistently maintained.

6. IANA Considerations

This document has no actions for IANA.

7. Security Considerations

Since the second-level name server has no knowledge of internal view data, internal resolvers must not use it directly for resolving queries. Only properly configured internal recursive forwarders must be approved to send queries to this name server and solely for the purpose of resolving external answers. DNS queries that are sent from the internal view name servers to the outside should only be destined to the second-level name server. The internal recursive forwarder must additionally not attempt to recursively answer queries if the authoritative name server for internal-view data fails to respond. If it did so, external data could be returned in such circumstances and lead to cache pollution.

The above examples are a quick reminder for the different ways in which it is possible to incorrectly configure the split-view DNS setup. Any misconfigurations in the three different types of name servers or the two packet filters can either result in cache pollution and cause incorrect results to be returned, or impede the ability for end resolvers to validate data returned in response to queries. An improperly configured packet filter that allows errant DNS traffic through or denies legitimate responses can lead to aggressive retransmission of queries.

Each of the validation options outlined in Section 3 also introduce their own security considerations. Using a common key between both

views of the split does not allow one to differentiate between internal and external data and troubleshooting is greatly encumbered. All approaches that use a common key for validating internal and external data are also susceptible to an insider attack where data from the outside view injected in response to internal queries can corrupt the cache. On the other hand, using a multitude of keys at end resolvers only increases the operator overhead and thus the chances for configuration errors.

8. Acknowledgements

The contributions, suggestions and remarks of the following persons to this draft are particularly acknowledged: Wesley Griffin, John Kelley, Russ Mundy and Sam Weiler. The two-level name server scheme described in this document builds upon work that was originally performed by Ed Lewis.

9. References

9.1 Normative References

[1] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "DNS Security Introduction and Requirements", Internet Draft ietf-dnsext-dnssec-intro, October 2004.

[2] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "Protocol Modifications for the DNS Security Extensions", Internet Draft ietf-dnsext-dnssec-proto, October 2004.

9.2 Informative References

[3] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", Internet Draft ietf-dnsop-bad-dns-res, July 2004.

Author's Address

Suresh Krishnaswamy
SPARTA Inc.
7075 Samuel Morse Dr.
Columbia, MD 21046
US

Email: suresh@tislabs.com
URI: http://www.sparta.com

Appendix A. Tracing the query flow

```
       ^                     root, com.
       |                     name servers
       |                          ^
    [Outside]                     |
       |                          +<--  --  --  -- -- -+
       v                          |                    |
   ------------[O u t e r----P a c k e t---F i l t e r--------------------
       ^                          |                    |
       |                          |                    v
       |                          |                   10.0.0.1
       |                          v                  example.com
   [10.0.0.0/8]            RECURSIVE NAMESERVER   sub.example.com(ext)
       |                       (10.0.0.2)
       |     CLIENT (10.0.0.3)     ^
       |                  ^        |
       v                  |        |
   ------------[I n n e r----P a c k e t---F i l t e r--------------------
       ^                  |        |
       |                  v        v
       |               INT RECURSIVE FORWARDER <---------> 192.168.2.3
       |                     (192.168.2.2)
   [192.168.0.0/16]                ^
       |                           |
       |                           v
       |                      192.168.2.1
       v                  sub.example.com(int)
```

Figure 15

Consider a fictitious domain "example.com" containing a delegation "sub.example.com", which is split into an internal (int) and an external view (ext). The name servers authoritative for the internal view data are located in the 192.168.0.0/16 network while the name servers authoritative for the external view is located in the boundary network (10.0.0.0/8). The name server at 10.0.0.1 is authoritative for both the example.com zone and the external view of sub.example.com. 10.0.0.3 is a DNS client configured to directly query the recursive forwarder at 192.168.2.1 for internal-view data.

[Note: while the above example lists example.com as residing with an RFC-1918 private-address space, in real-world situations, the boundary network would normally reside in a publically routable address space.]

The internal recursive forwarder 192.168.2.2 is configured to forward all queries for sub.example.com to the name server at 192.168.2.1

(which is authoritative for the inner view of this zone) and forwards all other queries to the recursive server at 10.0.0.2. Communication between 192.168.2.2 and 10.0.0.2 is protected using TSIG. 10.0.0.2 is configured to only answer those queries that are protected using the TSIG key that it shares with 192.168.2.2.

The packet filter between the internal network and boundary network only allows DNS queries from the internal network destined to 10.0.0.2, queries from 10.0.0.3 to the internal name servers and their corresponding responses, and denies all other DNS traffic. The packet filter between the external network and the boundary network allows all outbound queries from 10.0.0.2, all inbound queries to 10.0.0.1 and their respective responses while disallowing all other DNS traffic. There are no rules that affect forward and reverse traffic that result from queries that are sent by clients in the boundary network to authoritative external view name servers.

Unless stated otherwise, the "example." zone is assumed to contain the following delegation and glue:

```
sub.example.com         NS serv.sub.example.com
serv.sub.example.com    A  10.0.0.1
```

The following subsections trace the query and response flow that occurs in order to validate the query "a.sub.example.com/A" originating at a client on 192.168.2.3 and directed at 192.168.2.2, for each of the different approaches outlined in Section 3.

A.1 No validation

192.168.2.2 forwards the query to the authoritative internal-view name server (192.168.2.1), which returns an authoritative answer. No trusted key is configured at 192.168.2.2 so no validation of this response is done. The answer is returned to the client at 192.168.2.3 without any validation.

A.2 Same key signing

The example.com/DNSKEY record is configured as a trusted key at 192.168.2.2 and forms one end of the chain-of-trust. The delegations sub.example.com(ext) and sub.example.com(int) are signed using the same key; the DS record in example.com points to this key.

The answer for a.sub.example.com/A is returned from 192.168.2.1 as described in the previous section. 192.168.2.2 is able to build the chain-of-trust for a.sub.example.com/A from the following additional data -- sub.example.com/DNSKEY, sub.example.com/DS and example.com/DNSKEY.

The query for sub.example.com/DNSKEY is forwarded to the authoritative server at 192.168.2.1. Since sub.example.com/DS and example.com/DNSKEY are both external-view data, queries for them are forwarded by 192.168.2.2 to the second-level recursive server at 10.0.0.2.

192.168.2.2 determines the final answer based on all received responses and returns this to the client at 192.168.2.3.

A.3 Partial Decoupling of chains

The sub.example.com/DNSKEY (internal) record is configured as a trusted key at 192.168.2.2 and forms one end of the chain-of-trust.

The answer for a.sub.example.com/A is returned from 192.168.2.1 as described in the previous section. In order to complete the chain-of-trust for a.sub.example.com/A, 192.168.2.2 only needs to fetch the data for sub.example.com/DNSKEY.

It does this according to its forwarding rules by sending a query directly to the authoritative server for this data at 192.168.2.1, before finally returning the answer to the client at 192.168.2.3.

A.4 Complete Decoupling of chains

In this option, example.com is also split into an internal and external view.

[Note: For this sub-section, the name server at 192.168.2.1 is assumed to be authoritative for example.com(int) in addition to sub.example.com(int).

The delegation point for sub.example.com at example.com(int) is also assumed to contain glue that points to the name server at 192.168.2.1.

```
sub.example.com         NS serv.sub.example.com
serv.sub.example.com    A  192.168.2.1
```

The delegation point for sub.example.com at example.com(ext) contains glue that points to the name server at 10.0.0.1 and does not change.]

The configuration at 192.168.2.2 is modified to include a rule to forward queries for all data under example.com to 192.168.2.1, which is the name server authoritative for the internal view of this zone. The trusted key for both views of data is configured at the level of example.com but the exact key is different for the internal and the external view. The internal example.com/DNSKEY record is configured

as the trusted key at 192.168.2.2 and forms one end of the chain-of-trust.

The answer for a.sub.example.com/A is returned from 192.168.2.1 as before. In order to complete the chain-of-trust for a.sub.example.com/A, 192.168.2.2 validates the following data in succession -- sub.example.com/DNSKEY, sub.example.com/DS and example.com/DNSKEY. All of these queries are sent according to the forwarding rules in 192.168.2.2 to 192.168.2.1, which is the authoritative server for all of them.

192.168.2.2 determines the final answer based on all received responses and sends this to the client at 192.168.2.3.

A.5 Multiple DS records

sub.example.com(int) and sub.example.com(ext) are signed using different keys. The corresponding DS records are both present in example.com, which is not split. The trusted key is at the level of example.com and is the same for both views. This is configured as the trusted key at 192.168.2.2 and forms one end of the chain-of-trust.

The answer to a.sub.example.com/A is returned from 192.168.2.1 as before. In order to complete the chain-of-trust for a.sub.example.com/A, 192.168.2.2 validates the following data in succession -- sub.example.com/DNSKEY, sub.example.com/DS and example.com/DNSKEY.

The query for sub.example.com/DNSKEY is forwarded to the authoritative server at 192.168.2.1 while the the queries for sub.example.com/DS and example.com/DNSKEY are sent to the external recursive server at 10.0.0.2.

In constructing the chain-of-trust, the recursive server at 192.168.2.2 iteratively checks each DS record returned from the sub.example.com/DS and uses the one that successfully completes the chain for sub.example.com/DNSKEY (internal). 192.168.2.2 determines the final answer based on all reveived responses and sends this to the client at 192.168.2.3.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.