

Network Working Group
Internet-Draft
Expires: May 11, 2014

M. Andrews
ISC
November 7, 2013

Updating Parent Zones
draft-andrews-dnsop-update-parent-zones-04

Abstract

DNS UPDATE was developed to allow DNS zones to be updated.

There is a perception that UPDATE cannot be used in conjunction with the Registry, Registrar, Registrant (RRR) model to update a zone.

This document explains how UPDATE can be used in the RRR model.

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 May 11, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements	3
3. Translation	3
4. Authentication	4
5. Direct to Registrar	4
6. Indirect to Registrar	4
7. UPDATE Server Discovery	5
8. Security Considerations	6
9. Normative References	6
Author's Address	6

1. Introduction

UPDATE [RFC2136] is designed to update any zone in the DNS. This includes updating delegating NS records, glue address records and DS records.

While UPDATE is primarily designed to UPDATE a zone directly there is no reason why UPDATE requests cannot be translated to the EPP requests to perform the changes.

This would provide a uniform model to update parent zone regardless of where they are in the DNS hierarchy or whether the zone is signed or not.

2. Requirements

This document was written with the following requirements in mind:

- o must be able to authenticate the transaction.
- o must be able to update address records to support automated renumbering.
- o must be able to update DS records to support DNSKEY rollover but key management tools.
- o must work for unsigned zones (parent and/or child).
- o must work for signed zones (parent and/or child).
- o must work for RRR managed zones.
- o must work for non RRR managed zones.
- o desirable support updating of NS RRsets so that nameservers can ensure delegations delegation data remains consistent.

3. Translation

The Registrar would host a server that authenticates UPDATE requests received directly or relayed by the Registry using TSIG [RFC2845], then translate the actions in the UPDATE request into EPP transaction requests. The results of those EPP transactions would be relayed to the UPDATE client.

Requests that are not TSIG signed or fail verification must be rejected.

The translating server would handle a restricted subset of UPDATE requests, possibly ignoring the prerequisite section. UPDATE requests would be limited to those supported by EPP.

e.g. Add NS record. Delete all NS records. Add A record. Delete

AAAA record. Add DS record. Delete DS record.

The translating server may also override/ignore the TTL in the UPDATE request.

4. Authentication

Authentication would be done using TSIG. TSIG was designed to be used in a environment where requests are relayed.

Authentication can be done down to the <NAME,TYPE> tuple. There exist nameservers that already implement access controls down to this level of granularity based on the presented TSIG.

This would allow nameservers to update their own address records as they get renumbered without being able to update anything else.

This would allow DNSSEC key management software to update DS records without being able to update anything else.

As Registrars do all the authentication and generate the signed responses there is no need for the Registry to have access to the private key material used in TSIG.

Registrars already handle shared keys in these numbers with their web interfaces so it is not unreasonable to expect them to be able to handle a similar number of shared TSIG keys.

5. Direct to Registrar

The hardest part of Direct to Registrar is finding where to send the UPDATE request. This would most probably just be advised to the Registrant.

6. Indirect to Registrar

In the indirect model the Registry would host a UPDATE relay server which would examine the first record of the UPDATE section and relay the request to the Registrar of record for the owner name of that record. The Registrar would verify the validity of the request based on the TSIG then update the registry contents using EPP if appropriate. The response from the Registrar would be relayed back to the client via the Registry.

The Registry takes no action other than to relay the request and

response unless it is directed to do so by the Registrar.

The relay can use either TCP or UDP when forwarding UPDATE requests as TSIG supports changes to the DNS id field when a request/response is relayed. Only the Registrar and the client (Registrant) need to know the TSIG secret.

This is consistent with how tools like nsupdate work out where to send a UPDATE request if the zone is not explicitly set. They look at the ownername of the first record and use it to discover the containing zone.

7. UPDATE Server Discovery

UPDATE server discovery is a issue when the RRR model is in use as the UPDATE may need to be directed through EPP and/or sent to a Registrar. There are a number of way this could be done:

1) Adding a underscore infix labels to the zone which contain SRV records at pointing to Registrar/Registry servers for each child.

e.g. <child>._update._tcp.<parent> SRV 0 0 53 server.example.tld

The server pointed to could be be a relay server, as described above, or a UPDATE to EPP translating server. A relay server would allow for slower zone growth.

Using underscore infix labels requires no changes to nameservers operated by Registries but does require the zone content to be updated or a separate zone (e.g. _update._tcp.<parent>) to be delegated to contain this information.

A level of indirection could be added by using CNAME records to point to a domain operated by the registrar which contains the SRV record. This would allow the registrar to update the SRV records without having to update the zone being served by the registry. The CNAME would be updated on registrar changes. Note the target name the CNAME could also be managed by the registry as a way to consolidate the SRV record management.

```
child._update._tcp.tld CNAME registrar._registrars.tld
registrar._registrars.tld SRV 0 0 53 server.example.tld
```

As with traditional use of SRV, non-support can be signaled with

```
*._update._tcp SRV 0 0 0 .
```

If the Registry is operating a relay this can be supported with a single wildcard record.

```
*._update._tcp SRV 0 0 0 server.registry.tld
```

The client can fallback to direct update to parent servers if no SRV record is discovered. This allows the scheme to work outside of the registry, registrar, registrant model.

2) Extend UPDATE to return the update server. Currently the Zone section of the UPDATE refers to the zone to be update and is identified by the <QNAME,SOA,QCLASS> tuple. Replacing SOA with one or more of DS, NS, A and AAAA would allow a nameserver to distinguish between a traditional UPDATE request and a request to find the UPDATE servers. The tuple would contain the resource to be updated and the reply would contain SRV records pointing to the UPDATE servers. As there would possibly more than one parent the owner records would refer to the parent zone being updated.

8. Security Considerations

The UPDATE requests are all TSIG signed. This is a proven method for securing UPDATE requests in the DNS.

9. Normative References

- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

Author's Address

M. Andrews
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: marka@isc.org

IETF
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2015

L. Chapin, Ed.
Interisle Consulting Group
M. McFadden, Ed.
InterConnect Communications Ltd
March 2, 2015

Additional Reserved Top Level Domains
draft-chapin-additional-reserved-tlds-02

Abstract

The Internet Domain Name System (DNS) defines a tree of names starting with root, ".", immediately below which are top level domain (TLD) names such as ".com" and ".us". In June 1999 [RFC2606] reserved a small number of TLD names for use in documentation examples, private testing, experiments, and other circumstances in which it is desirable to avoid conflict with current or future actual TLD names in the DNS.

There has been significant evolution of Internet engineering and operation practices since [RFC2606] was published. In February 2013 [RFC6761] defined criteria and procedures for reserving a domain name for special use, and established an IANA registry for such names. This document reserves three domain name labels for special use in accordance with the criteria and procedures of [RFC6761]: home, corp, and mail.

It is important to note that TLD names may be reserved, in other contexts, for policy, political, or other reasons that are distinct from the IETF's concern with Internet engineering and operations. This document reserves TLD names only for operational and engineering reasons.

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 September 3, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	4
3. New top-level domain name reservations	4
4. Security Considerations	5
5. IANA Considerations	6
5.1. Domain Name Reservation Considerations for home	6
5.1.1. Users	6
5.1.2. Application Software	6
5.1.3. Name Resolution APSS and Libraries	6
5.1.4. Caching DNS Servers	7
5.1.5. Authoritative DNS Servers	7
5.1.6. DNS Server Operators	7
5.1.7. DNS Registries/Registrars	8
5.2. Domain Name Reservation Considerations for corp	8
5.2.1. Users	8
5.2.2. Application Software	8
5.2.3. Name Resolution APSS and Libraries	9
5.2.4. Caching DNS Servers	9
5.2.5. Authoritative DNS Servers	9
5.2.6. DNS Server Operators	9
5.2.7. DNS Registries/Registrars	10
5.3. Domain Name Reservation Considerations for mail	10
5.3.1. Users	10
5.3.2. Application Software	10
5.3.3. Name Resolution APSS and Libraries	11
5.3.4. Caching DNS Servers	11
5.3.5. Authoritative DNS Servers	11
5.3.6. DNS Server Operators	11
5.3.7. DNS Registries/Registrars	12

6. References	12
7. Acknowledgments	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Authors' Addresses	14

1. Introduction

The Internet Domain Name System is documented in [RFC1034], [RFC1035], [RFC1591] and numerous additional Requests for Comment. It defines a tree of names starting with root, ".", immediately below which are top level domain names such as ".com" and ".us". Below top level domain names there are normally additional levels of names.

[RFC2606] reserves a small number of TLD names which can be used for private testing of existing DNS related code, examples in documentation, DNS related experimentation, invalid DNS names, or other similar uses without fear of conflicts with current or future actual top-level domain names in the global DNS. [RFC2606] also notes that the Internet Assigned Numbers Authority (IANA) reserves the label "example" at the second level below the TLDs .com, .net, and .org.

Since [RFC2606] was published in 1999, Internet engineering and operation practices have evolved in ways that led to the publication in February 2013 of [RFC6761], which defined criteria and procedures for reserving a domain name for special use and established an IANA registry to which additional reserved special use names might be added as new requirements arose.

This document follows [RFC6761] to add three reserved top-level domain name labels to the IANA special-use names registry. It is prompted by the impending advent of new TLDs which might, in the absence of the reservations for which this document provides, introduce TLD labels that could create engineering and operational problems for root server operators and other DNS infrastructure providers.

It is important to note that TLD names may be reserved, in other contexts, for policy, political, or other reasons that are distinct from the IETF's concern with Internet engineering and operations. This document reserves TLD names only for operational and engineering reasons.

2. 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 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. New top-level domain name reservations

In its report [SAC045] of a quantitative study of queries to the DNS root servers entitled "Invalid Top Level Domain Queries at the Root Level of the Domain Name System" [SAC045] ICANN's Security and Stability Advisory Committee "calls attention to the potential problems that may arise should a new TLD applicant use a string that has been seen with measurable (and meaningful) frequency in a query for resolution by the root system and the root system has previously generated a response."

Of particular concern is the case in which a string "has been queried and a root name server has responded to the query with a non-existent domain (NXDOMAIN) result, i.e., the string has not been delegated but has been queried." [SAC045] reports the results of a CAIDA measurement study [RSSAC_DNS] which found that "NXDOMAIN responses account for more than 25 percent of the total responses from root name servers observed in the study, and the top ten such strings account for 10 percent of the total query load."

[SAC045] describes in detail the engineering and operational problems that would ensue from the delegation, as new valid TLD names, of previously invalid labels that have frequently appeared in queries to the root: "If the [new TLD label] were to be approved and the TLD included in the root zone, queries to the root level of the DNS for a string that hitherto returned NXDOMAIN would begin to return positive responses containing name servers of the new TLD."

Recommendation (2) of [SAC045] calls for the community to develop principles for "prohibiting the delegation of strings in addition to those already identified in [RFC2606]." As the first step in that process, based on the data reported by [SAC045], this document adds to the list of names that may not be used for top-level domains the following labels:

- o home
- o corp

These two top-level domain labels are to be added to the "Special-Use Domain Names" registry created by [RFC6761], as described in the IANA Considerations section of this document.

In addition, [SAC062] describes the risks associated with delegating a name in the root of the public DNS that is also used in privately defined namespaces (in which it is also syntactically valid). Users, software, or other functions in the private domain may confuse the private and public instances of the same name. This risk, referred to as "name collision," results in potential harm to enterprise networks that use previously undelegated names at the root of a private namespace when the name is delegated in the public root.

Research conducted by Interisle Consulting Group [INTERISLE] indicates that another name, in addition to those identified by [SAC045], presents a particularly high risk of name collision. This document therefore also adds the following string to the "Special-Use Domain Names" registry:

- o mail

Further resesarch, conducted by JAS Advisors on behalf of ICANN [JAS_MITIGATION] shows that the names .corp, .home and .mail are clear and significant risks for name collision. In that report the following recommendation is made: "The TLDs .corp, .home, and .mail be permanently reserved for internal use and receive RFC 1918-like protection/treatment, potentially via RFC 6761."

The three names that are reserved by this document are those on which all three studies (by SSAC, Interisle and JAS Advisors) agree.

4. Security Considerations

The name reservations specified in this document are intended to reduce the risk of harmful collision between names that are in well-established common use as TLDs in private namespaces and syntactically identical names that could otherwise be delegated as TLDs in the global DNS.

The security concerns associated with name collision are well presented in [SAC045], [SAC062], the Interisle report [INTERISLE], and the ICANN report "Name Collision Identification and Mitigation for IT Professionals" [ICANN_MITIGATION].

5. IANA Considerations

This document specifies three new labels to be added to the "Special-Use Domain Names" registry maintained by IANA pursuant to [RFC6761]. The labels are to be added to the registry in the following way:

Name	Reference
-----+-----	
home	[RFC-to-be]
corp	[RFC-to-be]
mail	[RFC-to-be]

Figure 1

5.1. Domain Name Reservation Considerations for home

5.1.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.1.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.1.3. Name Resolution APSs and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn't prevent them from using this reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these

names resolved locally rather than on the public Internet. Compliant name server software MUST NOT reject these names as invalid. Instead, name server software SHOULD allow for local resolution of the name and SHOULD NOT transmit a query for resolution into the public DNS.

5.1.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document MUST be denied.

5.2. Domain Name Reservation Considerations for corp

5.2.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.2.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.2.3. Name Resolution APSs and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn't prevent them from using this reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always

treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these names resolved locally rather than on the public Internet. Compliant name server software **MUST NOT** reject these names as invalid. Instead, name server software **SHOULD** allow for local resolution of the name and **SHOULD NOT** transmit a query for resolution into the public DNS.

5.2.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document **MUST** be denied.

5.3. Domain Name Reservation Considerations for mail

5.3.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.3.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.3.3. Name Resolution APSS and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn't prevent them from using this

reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these names resolved locally rather than on the public Internet. Compliant name server software MUST NOT reject these names as invalid. Instead, name server software SHOULD allow for local resolution of the name and SHOULD NOT transmit a query for resolution into the public DNS.

5.3.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document MUST be denied.

6. References

7. Acknowledgments

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

- [RFC1591] Postel, J., "Domain Name System Structure and Delegation", RFC 1591, March 1994.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

8.2. Informative References

- [ICANN_MITIGATION]
Internet Corporation for Assigned Names and Numbers,
"Guide to Name Collision Identification and Mitigation for
IT Professionals", August 2013,
<[http://www.icann.org/en/about/staff/security/ssr/
name-collision-mitigation-05dec13-en.pdf](http://www.icann.org/en/about/staff/security/ssr/name-collision-mitigation-05dec13-en.pdf)>.
- [INTERISLE]
Chapin, L., "Name Collision in the DNS", August 2013,
<[http://www.icann.org/en/about/staff/security/ssr/
name-collision-02aug13-en.pdf](http://www.icann.org/en/about/staff/security/ssr/name-collision-02aug13-en.pdf)>.
- [JAS_MITIGATION]
Internet Corporation for Assigned Names and Numbers,
"Mitigating the Risk of DNS Namespace Collisions",
February 2014,
<[https://www.icann.org/en/about/staff/security/ssr/name-
collision-mitigation-26feb14-en.pdf](https://www.icann.org/en/about/staff/security/ssr/name-collision-mitigation-26feb14-en.pdf)>.
- [RSSAC_DNS]
claffy, kc., "DNS Research Update from CAIDA Status and
Recent Experiences", March 2009,
<[http://www.caida.org/publications/presentations/2009/
rssac_dns/rssac_dns.pdf](http://www.caida.org/publications/presentations/2009/rssac_dns/rssac_dns.pdf)>.
- [SAC045] ICANN Security and Stability Advisory Committee, "Invalid
Top Level Domain Queries at the Root Level of the Domain
Name System", December 2010,
<<http://www.icann.org/en/committees/security/sac045.pdf>>.

[SAC062] ICANN Security and Stability Advisory Committee, "SSAC Advisory Concerning the Mitigation of Name Collision Risk", November 2013,
<<http://www.icann.org/en/groups/ssac/documents/sac-062-en.pdf>>.

Authors' Addresses

Lyman Chapin (editor)
Interisle Consulting Group
125A Magazine Street
Cambridge, MA 02139
UK

Phone: +1 617 686 2527
Email: lyman@interisle.net

Mark McFadden (editor)
InterConnect Communications Ltd
Merlin House; Station Road
Chepstow, Monmouthshire NP16 5PB
UK

Phone: +44 7792 276 904
Email: markmcfadden@icc-uk.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 1, 2014

A. Dulaunoy
CIRCL
A. Kaplan
CERT.at
P. Vixie
H. Stern
Farsight Security, Inc.
February 28, 2014

Passive DNS - Common Output Format
draft-dulaunoy-kaplan-passive-dns-cof-02

Abstract

This document describes a common output format of Passive DNS Servers which clients can query. The output format description includes also in addition a common semantic for each Passive DNS system. By having multiple Passive DNS Systems adhere to the same output format for queries, users of multiple Passive DNS servers will be able to combine result sets easily.

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 September 1, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Limitation	3
3. Common Output Format	3
3.1. Overview	3
3.2. ABNF grammar	4
3.3. Mandatory Fields	4
3.3.1. rrname	4
3.3.2. rrtype	5
3.3.3. rdata	5
3.3.4. time_first	5
3.3.5. time_last	6
3.4. Optional Fields	6
3.4.1. count	6
3.4.2. bailiwick	6
3.5. Additional Fields	6
3.5.1. sensor_id	6
3.5.2. zone_time_first	6
3.5.3. zone_time_last	7
3.6. Additional Fields Registry	7
4. Acknowledgements	7
5. IANA Considerations	7
6. Privacy Considerations	7
7. Security Considerations	7
8. References	8
8.1. Normative References	8
8.2. References	8
8.3. Informative References	9
Appendix A. Examples	10
Authors' Addresses	10

1. Introduction

Passive DNS is a technique described by Florian Weimer in 2005 in Passive DNS replication, F Weimer - 17th Annual FIRST Conference on Computer Security [WEIMERPDNS]. Since then multiple Passive DNS implementations were created and evolved over time. Users of these Passive DNS servers may query a server (often via WHOIS [RFC3912] or HTTP REST [REST]), parse the results and process them in other applications.

There are multiple implementations of Passive DNS software. Users of passive DNS query each implementation and aggregate the results for their search. This document describes the output format of four Passive DNS Systems ([DNSDB], [PDNSCERTAT], [PDNSCIRCL] and [PDNSCOF]) which are in use today and which already share a nearly identical output format. As the format and the meaning of output fields from each Passive DNS need to be consistent, we propose in this document a solution to commonly name each field along with their corresponding interpretation. The format follows a simple key-value structure in JSON [RFC4627] format. The benefit of having a consistent Passive DNS output format is that multiple client implementations can query different servers without having to have a separate parser for each individual server. `passivedns-client` [PDNSCLIENT] currently implements multiple parsers due to a lack of standardization. The document does not describe the protocol (e.g. WHOIS [RFC3912], HTTP REST [REST]) nor the query format used to query the Passive DNS. Neither does this document describe "pre-recursor" Passive DNS Systems. Both of these are separate topics and deserve their own RFC document.

1.1. 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].

2. Limitation

As a Passive DNS servers can include protection mechanisms for their operation, results might be different due to those protection measures. These mechanisms filter out DNS answers if they fail some criteria. The bailiwick algorithm [BAILIWICK] protects the Passive DNS Database from cache poisoning attacks [CACHEPOISONING]. Another limitation that clients querying the database need to be aware of is that each query simply gets a snapshot-answer of the time of querying. Clients MUST NOT rely on consistent answers. Nor must they assume that answers must be identical across multiple Passive DNS Servers.

3. Common Output Format

3.1. Overview

The formatting of the answer follows the JSON [RFC4627] format. In fact, it is a subset of the full JSON language. Notable differences are the modified definition of whitespace ("ws"). The order of the fields is not significant for the same resource type.

The intent of this output format is to be easily parsable by scripts. Each JSON object is expressed on a single line to be processed by the client line-by-line. Every implementation **MUST** support the JSON output format.

Examples of JSON (Appendix A) output are in the appendix.

3.2. ABNF grammar

Formal grammar as defined in ABNF [RFC2234]

```

answer           = entries
entries          = * ( entry CR)
entry            = "{" keyvallist "}"
keyvallist       = [ member *( value-separator member ) ]
member           = qm field qm name-separator value
name-separator   = ws %x3A ws           ; a ":" colon
value            = value                ; as defined in the JSON RFC
value-separator  = ws %x2C ws           ; , comma. As defined in JSON
field            = "rrname" | "rrtype" | "rdata" | "time_first" |
                  "time_last" | "count" | "bailiwick" | "sensor_id" |
                  "zone_time_first" | "zone_time_last" | futureField
futureField      = string
CR               = %x0D
qm               = %x22                 ; " a quotation mark
ws               = *(
                  %x20 |                 ; Space
                  %x09                   ; Horizontal tab
                  )

```

Note that value is defined in JSON [RFC4627] and has the exact same specification as there. The same goes for the definition of string.

3.3. Mandatory Fields

Implementation **MUST** support all the mandatory fields.

Uniqueness property: the tuple (rrname,rrtype,rdata) will always be unique within one answer per server. While rrname and rrtype are always individual JSON primitive types (strings, numbers, booleans or null), rdata MAY return multiple resource records or a single record. When multiple resource records are returned, rdata **MUST** be a JSON array. In the case of a single resource record is returned, rdata **MUST** be a JSON string.

3.3.1. rrname

This field returns the name of the queried resource.

3.3.2. rrtype

This field returns the resource record type as seen by the passive DNS. The key is rrtype and the value is in the interpreted record type represented as a JSON [RFC4627] string. If the value cannot be interpreted the decimal value is returned following the principle of transparency as described in RFC 3597 [RFC3597]. Then the decimal value is represented as a JSON [RFC4627] number. The resource record type can be any values as described by IANA in the DNS parameters document in the section 'Resource Record (RR) TYPES' (<http://www.iana.org/assignments/dns-parameters>). Currently known and supported textual descriptions of rrtypes are: A, AAAA, CNAME, PTR, SOA, TXT, DNAME, NS, SRV, RP, NAPTR, HINFO, A6. A client MUST be able to understand these textual rrtype values represented as a JSON [RFC4627] string. In addition, a client MUST be able to handle a decimal value (as mentioned above) as answer represented as a JSON [RFC4627] number.

3.3.3. rdata

This field returns the resource records of the queried resource. When multiple resource records are returned, rdata MUST be a JSON array. In the case of a single resource record is returned, rdata MUST be a JSON string. Each resource record is represented as a JSON [RFC4627] string. Each resource record MUST be escaped as defined in section 2.6 of RFC4627 [RFC4627]. Depending on the rrtype, this can be an IPv4 or IPv6 address, a domain name (as in the case of CNAMEs), an SPF record, etc. A client MUST be able to interpret any value which is legal as the right hand side in a DNS master file RFC 1035 [RFC1035] and RFC 1034 [RFC1034]. If the rdata came from an unknown DNS resource records, the server must follow the transparency principle as described in RFC 3597 [RFC3597].

3.3.4. time_first

This field returns the first time that the record / unique tuple (rrname, rrtype, rdata) has been seen by the passive DNS. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.3.5. time_last

This field returns the last time that the unique tuple (rrname, rrtype, rdata) record has been seen by the passive DNS. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.4. Optional Fields

Implementations SHOULD support one or more fields.

3.4.1. count

Specifies how many authoritative DNS answers were received at the Passive DNS Server's collectors with exactly the given set of values as answers (i.e. same data in the answer set - compare with the uniqueness property in "Mandatory Fields"). The number of requests is expressed as a decimal value. This field is represented as a JSON [RFC4627] number.

3.4.2. bailiwick

The bailiwick is the best estimate of the apex of the zone where this data is authoritative.

3.5. Additional Fields

Implementations MAY support the following fields:

3.5.1. sensor_id

This field returns the sensor information where the record was seen. It is represented as a JSON [RFC4627] string.

3.5.2. zone_time_first

This field returns the first time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.5.3. zone_time_last

This field returns the last time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import. The date is expressed in seconds (decimal) since 1st of January 1970 (Unix timestamp). The time zone MUST be UTC. This field is represented as a JSON [RFC4627] number.

3.6. Additional Fields Registry

In accordance with [RFC6648], designers of new passive DNS applications that would need additional fields can request and register new field name at <https://github.com/adulau/pdns-qof/wiki/Additional-Fields>.

4. Acknowledgements

Thanks to the Passive DNS developers who contributed to the document.

5. IANA Considerations

This memo includes no request to IANA.

6. Privacy Considerations

Passive DNS Servers capture DNS answers from multiple collecting points ("sensors") which are located on the Internet-facing side of DNS recursors ("post-recursor passive DNS"). In this process, they intentionally omit the source IP, source port, destination IP and destination port from the captured packets. Since the data is captured "post-recursor", the timing information (who queries what) is lost, since the recursor will cache the results. Furthermore, since multiple sensors feed into a passive DNS server, the resulting data gets mixed together, reducing the likelihood that Passive DNS Servers are able to find out much about the actual person querying the DNS records nor who actually sent the query. In this sense, passive DNS Servers are similar to keeping an archive of all previous phone books - if public DNS records can be compared to phone numbers - as they often are. Nevertheless, the authors strongly encourage Passive DNS implementors to take special care of privacy issues. [bortzmeyer-dnsop-dns-privacy](#) is an excellent starting point for this. Finally, the overall recommendations in RFC6973 [RFC6973] should be taken into consideration when designing any application which uses Passive DNS data.

7. Security Considerations

In some cases, Passive DNS output might contain confidential information and its access might be restricted. When a user is querying multiple Passive DNS and aggregating the data, the sensitivity of the data must be considered.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.
- [RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, September 2004.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC5001] Austein, R., "DNS Name Server Identifier (NSID) Option", RFC 5001, August 2007.
- [RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, June 2012.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

8.2. References

- [BAILIWICK] "Passive DNS Hardening", 2010, <https://archive.farsightsecurity.com/Passive_DNS/passive_dns_hardening_handout.pdf>.

- [CACHEPOISONING]
"Black ops 2008: It's the end of the cache as we know it.", 2008, <http://kurser.lobner.dk/dDist/DMK_BO2K8.pdf>.
- [DNSDB] "DNSDB API", 2013, <<https://api.dnsdb.info/>>.
- [PDNSCERTAT]
"pDNS presentation at 4th Centr R&D workshop Frankfurt Jun 5th 2012", 2012, <http://www.centri.org/system/files/agenda/attachment/rd4-papst-passive_dns.pdf>.
- [PDNSCIRCL]
"CIRCL Passive DNS", 2012, <<http://pdns.circl.lu/>>.
- [PDNSCLIENT]
"Queries 5 major Passive DNS databases: BFK, CERTEE, DNSParse, ISC, and VirusTotal.", 2013, <<https://github.com/chrislee35/passivedns-client>>.
- [PDNSCOF] "Passive DNS server interface using the common output format", 2013, <<https://github.com/adulau/pdns-qof-server/>>.
- [REST] "Representational State Transfer (REST)", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.
- [WEIMERPDNS]
"Passive DNS Replication", 2005, <<http://www.enyo.de/fw/software/dnslogger/first2005-paper.pdf>>.

8.3. Informative References

- [I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Appendix A. Examples

The JSON output are represented on multiple lines for readability but each JSON object should on a single line.

If you query a passive DNS for the rrname `www.ietf.org`, the passive dns common output format can be:

```
{"count": 102, "time_first": 1298412391, "rrtype": "AAAA",  
"rrname": "www.ietf.org", "rdata": "2001:1890:1112:1::20",  
"time_last": 1302506851}  
{"count": 59, "time_first": 1384865833, "rrtype": "A",  
"rrname": "www.ietf.org", "rdata": "4.31.198.44",  
"time_last": 1389022219}
```

If you query a passive DNS for the rrname `ietf.org`, the passive dns common output format can be:

```
{"count": 109877, "time_first": 1298398002, "rrtype": "NS",  
"rrname": "ietf.org", "rdata": "ns1.yyz1.afiliast.net",  
"time_last": 1389095375}  
{"count": 4, "time_first": 1298495035, "rrtype": "A",  
"rrname": "ietf.org", "rdata": "64.170.98.32",  
"time_last": 1298495035}  
{"count": 9, "time_first": 1317037550, "rrtype": "AAAA",  
"rrname": "ietf.org", "rdata": "2001:1890:123a::1:1e",  
"time_last": 1330209752}
```

Please note that in the examples above, any backslashes `"\"` can be ignored and are an artefact of the tools which produced this document.

Authors' Addresses

Alexandre Dulaunoy
CIRCL
41, avenue de la gare
Luxembourg L-1611
Luxembourg

Phone: (+352) 247 88444
Email: alexandre.dulaunoy@circl.lu
URI: <http://www.circl.lu/>

L. Aaron Kaplan
CERT.at
Karlsplatz 1/2/9
Vienna A-1010
Austria

Phone: +43 1 5056416 78
Email: kaplan@cert.at
URI: <http://www.cert.at/>

Paul Vixie
Farsight Security, Inc.
11400 La Honda Road
Woodside, California 94062
U.S.A.

Email: paul@redbarn.org
URI: <https://www.farsightsecurity.com/>

Henry Stern
Farsight Security, Inc.
11400 La Honda Road
Woodside, California 94062
U.S.A.

Phone: +1 650 542-7836
Email: henry@stern.ca
URI: <https://www.farsightsecurity.com/>

INTERNET-DRAFT
Intended Status: Proposed Standard

Donald Eastlake
Huawei
Mark Andrews
ISC
October 11, 2014

Expires: April 10, 2015

Domain Name System (DNS) Cookies
<draft-eastlake-dnsext-cookies-05.txt>

Abstract

DNS cookies are a lightweight DNS transaction security mechanism that provides limited protection to DNS servers and clients against a variety of increasingly common denial-of-service and amplification / forgery or cache poisoning attacks by off-path attackers. DNS Cookies are tolerant of NAT, NAT-PT, and anycast and can be incrementally deployed.

Status of This Document

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

Distribution of this document is unlimited. Comments should be sent to the author or the DNSEXT mailing list <dnsext@ietf.org>.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>. The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Table of Contents

1. Introduction.....	3
1.1 Contents of This Document.....	3
1.2 Definitions.....	4
2. Threats Considered.....	5
2.1 Denial-of-Service Attacks.....	5
2.1.1 DNS Amplification Attacks.....	5
2.1.2 DNS Server Denial-of-Service.....	5
2.2 Cache Poisoning and Answer Forgery Attacks.....	6
3. Comments on Existing DNS Security.....	7
3.1 Existing DNS Data Security.....	7
3.2 DNS Message/Transaction Security.....	7
3.3 Conclusions on Existing DNS Security.....	7
4. The COOKIE OPT Option.....	8
4.1 Client Cookie.....	9
4.2 Server Cookie.....	9
4.3 Error Code.....	10
5. DNS Cookies Protocol Description.....	11
5.1 Originating Requests.....	11
5.2 Responding to Requests.....	11
5.2.1 No OPT RR.....	12
5.2.2 No Valid Client Cookie.....	12
5.2.3 Bad or Absent Server Cookie.....	13
5.2.4 A Correct Server Cookie.....	13
5.3 Processing Responses.....	14
5.4 Client and Server Secret Rollover.....	14
5.5 Implementation Requirement.....	15
6. NAT Considerations and AnyCast Server Considerations...	16
7. Deployment.....	18
8. IANA Considerations.....	19
9. Security Considerations.....	20
9.1 Cookie Algorithm Considerations.....	20
Acknowledgements.....	21
Normative References.....	22
Informative References.....	22
Appendix A: Example Client Cookie Algorithms.....	24
A.1 A Simple Algorithm.....	24
A.2 A More Complex Algorithm.....	24
Appendix B: Example Server Cookie Algorithms.....	25
B.1 A Simple Algorithm.....	25
B.2 A More Complex Algorithm.....	25

1. Introduction

As with many core Internet protocols, the Domain Name System (DNS) was originally designed at a time when the Internet had only a small pool of trusted users. As the Internet has grown exponentially to a global information utility, the DNS has increasingly been subject to abuse.

This document describes DNS cookies, a lightweight DNS transaction security mechanism specified as an OPT [RFC6891] option. This mechanism provides limited protection to DNS servers and clients against a variety of increasingly common abuses by off-path attackers. It is compatible with and can be used in conjunction with other DNS transaction forgery resistance measures such as those in [RFC5452].

The protection provided by DNS cookies bears some resemblance to that provided by using TCP for DNS transactions. To bypass the weak protection provided by using TCP requires an off-path attacker guessing the 32-bit TCP sequence number in use. To bypass the weak protection provided by DNS Cookies requires such an attacker to guess a 64-bit pseudo-random quantity. Where DNS Cookies are not available but TCP is, a fall back to using TCP is a reasonable strategy.

If only one party to a DNS transaction supports DNS cookies, the mechanism does not provide a benefit or significantly interfere; but, if both support it, the additional security provided is automatically available.

The DNS cookies mechanism is designed to work in the presence of NAT and NAT-PT boxes and guidance is provided herein on supporting the DNS cookies mechanism in anycast servers.

1.1 Contents of This Document

In Section 2, we discuss the threats against which the DNS cookie mechanism provides some protection.

Section 3 describes existing DNS security mechanisms and why they are not adequate substitutes for DNS cookies.

Section 4 describes the COOKIE OPT option.

Section 5 provides a protocol description.

Section 6 discusses some NAT and anycast related DNS Cookies design considerations.

Section 7 discusses incremental deployment considerations.

Sections 8 and 9 describe IANA and Security Considerations.

1.2 Definitions

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 [RFC2119].

An "off-path attacker", for a particular DNS client and server, is defined as an attacker who cannot observe the DNS request and response messages between that client and server.

"Soft state" indicates information learned or derived by a host which may be discarded when indicated by the policies of that host but can be later re-instantiated if needed. For example, it could be discarded after a period of time or when storage for caching such data becomes full. If operations requiring that soft state continue after it has been discarded, it will be automatically re-generated, albeit at some cost.

"Silently discarded" indicates that there are no DNS protocol message consequences; however, it is RECOMMENDED that appropriate network management facilities be included in implementations, such as a counter of the occurrences of each such event type.

"IP address" is used herein as a length independent term and includes both IPv4 and IPv6 addresses.

2. Threats Considered

DNS cookies are intended to provide significant but limited protection against certain attacks by off-path attackers as described below. These attacks include denial-of-service, cache poisoning and answer forgery.

2.1 Denial-of-Service Attacks

The typical form of the denial-of-service attacks considered herein is to send DNS requests with forged source IP addresses to a server. The intent can be to attack that server or some other selected host as described below.

2.1.1 DNS Amplification Attacks

A request with a forged IP address generally causes a response to be sent to that forged IP address. Thus the forging of many such requests with a particular source IP address can result in enough traffic being sent to the forged IP address to interfere with service to the host at the IP address. Furthermore, it is generally easy in the DNS to create short requests that produce much longer responses, thus amplifying the attack.

The DNS Cookies mechanism can severely limit the traffic amplification obtained by attackers off path for the server and the attacked host. Enforced DNS cookies would make it hard for an off path attacker to cause any more than rate-limited short error responses to be sent to a forged IP address so the attack would be attenuated rather than amplified. DNS cookies make it more effective to implement a rate limiting scheme for error responses from the server. Such a scheme would further restrict selected host denial-of-service traffic from that server.

2.1.2 DNS Server Denial-of-Service

DNS requests that are accepted cause work on the part of DNS servers. This is particularly true for recursive servers that may issue one or more requests and process the responses thereto, in order to determine their response to the initial request. And the situation can be even worse for recursive servers implementing DNSSEC ([RFC4033] [RFC4034] [RFC4035]) because they may be induced to perform burdensome cryptographic computations in attempts to verify the authenticity of data they retrieve in trying to answer the

request.

The computational or communications burden caused by such requests may not dependent on a forged IP source address, but the use of such addresses makes

- + the source of the requests causing the denial-of-service attack harder to find and
- + restriction of the IP addresses from which such requests should be honored hard or impossible to specify or verify.

Use of DNS cookies should enables a server to reject forged queries from an off path attacker with relative ease and before any recursive queries or public key cryptographic operations are performed.

2.2 Cache Poisoning and Answer Forgery Attacks

The form of the cache poisoning attacks considered is to send forged replies to a resolver. Modern network speeds for well-connected hosts are such that, by forging replies from the IP addresses of a DNS server to a resolver for common names or names that resolver has been induced to resolve, there can be an unacceptably high probability of randomly coming up with a reply that will be accepted and cause false DNS information to be cached by that resolver (the Dan Kaminsky attack). This can be used to facilitate phishing attacks and other diversion of legitimate traffic to a compromised or malicious host such as a web server.

With the use of DNS cookies, a resolver can generally reject such forged replies.

3. Comments on Existing DNS Security

Two forms of security have been added to DNS, data security and message/transaction security.

3.1 Existing DNS Data Security

DNS data security is one part of DNSSEC and is described in [RFC4033], [RFC4034], and [RFC4035] and updates thereto. It provides data origin authentication and authenticated denial of existence. DNSSEC is being deployed and can provide strong protection against forged data; however, it has the unintended effect of making some denial-of-service attacks worse because of the cryptographic computational load it can require and the increased size in DNS response packets that it tends to produce.

3.2 DNS Message/Transaction Security

The second form of security that has been added to DNS provides "transaction" security through TSIG [RFC2845] or SIG(0) [RFC2931]. TSIG could provide strong protection against the attacks for which the DNS Cookies mechanism provide weak protection; however, TSIG is non-trivial to deploy in the general Internet because of the burden it imposes of pre-agreement and key distribution between client-server pairs, the burden of server side key state, and because it requires time synchronization between client and server.

TKEY [RFC2930] can solve the problem of key distribution for TSIG but some modes of TKEY impose a substantial cryptographic computation loads and can be dependent on the deployment of DNS data security (see Section 3.1).

SIG(0) [RFC2931] provides less denial of service protection than TSIG or, in one way, even DNS cookies, because it does not authenticate requests, only complete transactions. In any case, it also depends on the deployment of DNS data security and requires computationally burdensome public key cryptographic operations.

3.3 Conclusions on Existing DNS Security

The existing DNS security mechanisms do not provide the services provided by the DNS Cookies mechanism: lightweight message authentication of DNS requests and responses with no requirement for pre-configuration or per client server side state.

4. The COOKIE OPT Option

COOKIE is an OPT RR [RFC6891] option that can be included in the RDATA portion of an OPT RR in DNS requests and responses. The option length varies depending on the circumstance in which it is being used. There are two cases as described below. Both use the same OPTION-CODE; they are distinguished by their length.

In a request sent by a client to a server when the client does not know the server cookie, its length is 10, consisting of a 2 bytes DNS error code field followed by the 8 byte Client Cookie as shown in Figure 1.

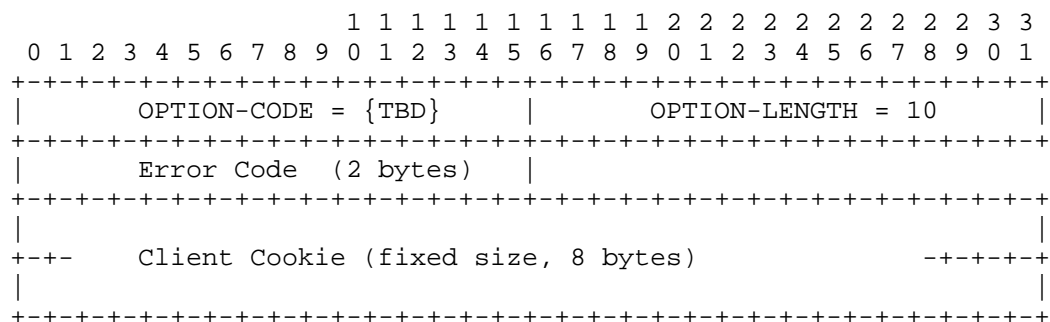


Figure 1. COOKIE Option, Unknown Server Cookie

In a request sent by a client when a server cookie is known and in all responses, the length is variable from 18 to 42 bytes, consisting of a 2 byte DNS error field followed by the 8 bytes Client Cookie and then the variable 8 to 32 bytes Server Cookie as shown in Figure 2. The variability of the option length stems from the variable length Server Cookie. The Server Cookie is an integer number of bytes with a minimum size of 64 bits for security and a maximum size of 256 bits for implementation convenience.

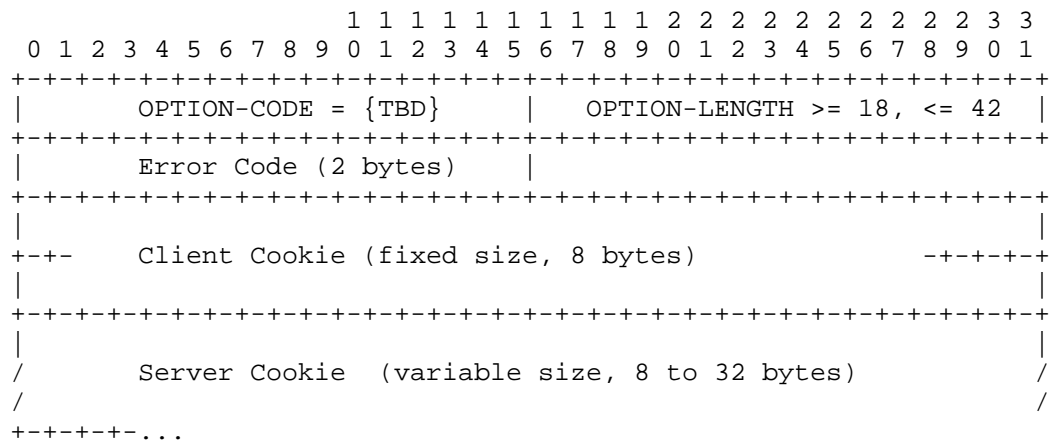


Figure 2. COOKIE Option, Known Server Cookie

4.1 Client Cookie

The Client Cookie SHOULD be a pseudo-random function of the server IP address and a secret quantity known only to the client. This client secret SHOULD have at least 64 bits of entropy [RFC4086] and be changed periodically (see Section 5.4). The selection of the pseudo-random function is a matter private to the client as only the client needs to recognize its own DNS cookies.

For further discussion of the Client Cookie field, see Section 5.1. For example methods of determining a Client Cookie, see Appendix A.

A client MUST NOT use the same Client Cookie value for queries to all servers.

4.2 Server Cookie

The Server Cookie SHOULD consist of or include a 64-bit or larger pseudo-random function of the request source IP address, the request Client Cookie, and a secret quantity known only to the server. (See Section 6 for a discussion of why the Client Cookie is used as input to the Server Cookie but the Server Cookie is not used as an input to the Client Cookie.) This server secret SHOULD have at least 64 bits of entropy [RFC4086] and be changed periodically (see Section 5.4). The selection of the pseudo-random function is a matter private to the server as only the server needs to recognize its own DNS cookies.

For further discussion of the Server Cookie field see Section 5.2.
For example methods of determining a Server Cookie, see Appendix B.

A server MUST NOT use the same Server Cookie value for responses to all clients.

4.3 Error Code

In requests, the Error Code field MUST be zero and is ignored on receipt. Replies have a COOKIE OPT with an Error Code equal to one of the following four values: Zero (if the request they respond to had a COOKIE OPT with a correct Server Cookie), NOCOOKIE, MFCOOKIE, or BADCOOKIE.

NOCOOKIE and MFCOOKIE indicate that the server did not receive a Client Cookie, either because there was no COOKIE OPT option in the request (NOCOOKIE) or one was present but the COOKIE OPT option was malformed as not being a valid length (MFCOOKIE). BADCOOKIE indicates that the server did receive a Client Cookie but did not receive the correct Server Cookie either because there was no Server Cookie present or because it was not a valid value.

A server may choose to normally process a request, for example returning the normal answer information for a QUERY, notwithstanding a cookie error condition. For more information on error processing, see Section 5.

5. DNS Cookies Protocol Description

This section discusses using DNS Cookies in the DNS Protocol.

5.1 Originating Requests

A DNS client that implements DNS cookies includes one DNS Cookie option in every DNS request it sends unless DNS cookies are disabled. The COOKIE OPT option in a request always includes a zero Error Code field and a Client Cookie as discussed in Section 4.1.

If the client has no Server Cookie obtained from a previous DNS response and cached under the server's IP address, it uses the shorter form of COOKIE OPT shown in Figure 1. If the client does have such a cached Server Cookie, it uses the form of COOKIE OPT shown in Figure 2 and also includes that cached Server Cookie in the DNS option it sends.

5.2 Responding to Requests

The Server Cookie, when it occurs in a COOKIE OPT option in a request, is intended to weakly assure the server that the request came from a client that is both at the source IP address of the request and using the Client Cookie included in the option. This weak assurance is provided by the Server Cookie that server would send to that client in an earlier response appearing as the Server Cookie field in the request.

At a server where DNS Cookies are not implemented and enabled, the presence of a COOKIE OPT option is ignored and the server responds as before.

When DNS Cookies are implemented and enabled, there are four possibilities: (1) there is no OPT RR at all in the request; (2) there is no valid Client Cookie in the request because the COOKIE OPT option is absent from the request or one is present but not a legal length; (3) there is a valid length cookie option in the request with no Server Cookie or an incorrect Server Cookie; or (4) there is a cookie option in the request with a correct Server Cookie. The four possibilities are discussed in the subsections below.

In the case of multiple COOKIE OPT options in a request, only the first (the one closest to the DNS header) is considered. All others are ignored.

5.2.1 No OPT RR

If there is no OPT RR in the request, the client does not support EDNS since [RFC6891] requires that an OPT RR be included in a request if the requester supports that feature. Under these circumstances, the server cannot expect to ever receive a correct COOKIE OPT option from the client as in Section 5.2.4.

The situation and server options available are the same as those in Section 5.2.2 except that no OPT RR can be included in any response.

5.2.2 No Valid Client Cookie

A request with an OPT RR but no COOKIE OPT option or with a COOKIE that is not a valid length (10 or 18 through 42) could be from a client that does not implement DNS cookies or on which they are disabled or it could be some form of abuse or broken client implementation. A server on which DNS cookies are enabled has the following three choices in responding to such a request:

- (1) Silently discard the request.
- (2) Not process the request other than returning a minimal length error response. Because of the absence of a validly formatted COOKIE OPT option in the request, it cannot be assumed that the client would understand any new RCODE values. An RCODE of Refused is returned and the Error Field of the returned COOKIE OPT option is set to NOCOOKIE if there was no COOKIE OPT option in the request and set to MFCOOKIE if such an option was present but not a valid length.
- (3) Process the request normally and provide a normal response except that a COOKIE OPT option with a non-zero Error Field is included as in point 2 above. The RCODE in the DNS Header is zero unless some non-cookie error occurs in processing the request.

Server policy determines how often the server selects each of the above response choices; however, if the request was received over TCP, the server may wish to take the weak authentication provided by the use of TCP into account, increasing the probability of choice 3 and decreasing the probability of choice 1 perhaps to the extent of never choosing 1. For both response choices 2 and 3, the server should consider setting TC=1 in the response so that future requests from the client are more likely to be received with the weak authentication that can be provided by TCP.

5.2.3 Bad or Absent Server Cookie

If a request is received with the COOKIE OPT option having no Server Cookie value (length 10) or a bad Server Cookie value (length 18 to 42), it could be some attempted abuse or it could just be that the client does not know a currently valid Server Cookie for the server to which the request was sent. For example, the client might have an old, no longer recognized Server Cookie

Servers MUST, at least occasionally, respond to such requests to inform the client of the correct Server Cookie. This is necessary so that such a client can bootstrap to the weakly secure state where requests and responses have recognized Server Cookies and Client Cookies.

In responding to such a request, the server has the following three choices:

- (1) Silently discard the request.
- (2) Not process the request other than returning a minimal length error response. Because of the correct length COOKIE OPT option in the request, the client can be assume to understand the new error codes assigned in this document. Both the Error Field in the returned COOKIE OPT option and the extended RCODE are set to BADCOOKIE.
- (3) Processes the request normally and sends its usual response including a COOKIE OPT option with an Error field of BADCOOKIE and a zero RCODE (unless there was also a non-cookie error in processing the request).

Server policy determines how often the server selects each of the above response choices; however, if the request was received over TCP, the server may wish to take the weak authentication provided by the use of TCP into account, increasing the probability of choice 3 and decreasing the probability of choice 1 perhaps to the extent of never choosing 1.

5.2.4 A Correct Server Cookie

If a server with enabled DNS cookies receives a request where the COOKIE OPT option has a valid length and correct Server Cookie, it processes the request normally and includes a COOKIE OPT option with a zero Error Field in the response. Such a response might have a non-zero RCODE if a non-cookie error occurs in processing the request.

5.3 Processing Responses

The Client Cookie, when it occurs in a COOKIE OPT option in a DNS reply, is intended to weakly assure the client that the reply came from a server at the source IP address use in the response packet because the Client Cookie value is the value that client would send to that server in a request. If there are multiple COOKIE OPT options in a DNS reply, all but the first (the one closest to the DNS Header) are ignored.

A DNS client where DNS cookies are implemented and enabled examines response for DNS cookies and MUST discard the response if it contains an illegal COOKIE OPT option length or an incorrect Client Cookie value. If the COOKIE OPT option Client Cookie is correct, the client caches the Server Cookie provided even if the response is an error response (RCODE non-zero).

If the reply extended RCODE is BADCOOKIE, it means that the server was unwilling to process the request because it did not have the correct Server Cookie in it. The client should retry the request using the new Server Cookie from the response.

If the RCODE is some value other than BADCOOKIE, including zero, the response is then processed normally.

5.4 Client and Server Secret Rollover

Clients and servers MUST NOT continue to use the same secret in new queries and responses, respectively, for more than 14 days and SHOULD NOT continue to do so for more than 1 day. Many clients rolling over their secret at the same time could briefly increase server traffic and exactly predictable rollover times for clients or servers might facilitate guessing attacks. For example, an attacker might increase the priority of attacking secrets they believe will be in effect for an extended period of time. To avoid rollover synchronization and predictability, it is RECOMMENDED that pseudorandom jitter of at least 30% be applied to the time of a scheduled rollover of a DNS cookie secret.

It is RECOMMENDED that a client keep the Client Cookie it is expecting in a reply associated with the outstanding query to avoid rejection of replies due to a bad Client Cookie right after a change in the Client Secret. It is RECOMMENDED that a server retain its previous secret for a period of time not less than 1 second or more than 3 minutes, after a change in its secret, and consider queries with Server Cookies based on its previous secret to have a correct Server Cookie during that time.

Receiving a sudden increased level of requests with bad Server Cookies or replies with bad Client Cookies would be a good reason to believe a server or client is likely to be under attack and should consider more frequent rollover of its secret.

5.5 Implementation Requirement

DNS clients and servers SHOULD implement DNS cookies to decrease their vulnerability to the threats discussed in Section 2.

6. NAT Considerations and AnyCast Server Considerations

In the Classic Internet, DNS Cookies could simply be a pseudo-random function of the client IP address and a sever secret or the server IP address and a client secret. You would want to compute the Server Cookie that way, so a client could cache its Server Cookie for a particular server for an indefinitely amount of time and the server could easily regenerate and check it. You could consider the Client Cookie to be a weak client signature over the server IP address that the client checks in replies and you could extend this weak signature to cover the request ID, for example, or any other information that is returned unchanged in the reply.

But we have this reality called NAT [RFC3022], Network Address Translation (including, for the purposes of this document, NAT-PT, Network Address and Protocol Translation, which has been declared Historic [RFC4966]). There is no problem with DNS transactions between clients and servers behind a NAT box using local IP addresses. Nor is there a problem with NAT translation of internal addresses to external addresses or translations between IPv4 and IPv6 addresses, as long as the address mapping is relatively stable. Should the external IP address an internal client is being mapped to change occasionally, the disruption is little more than when a client rolls-over its DNS COOKIE secret. And normally external access to a DNS server behind a NAT box is handled by a fixed mapping which forwards externally received DNS requests to a specific host.

However, NAT devices sometimes also map ports. This can cause multiple DNS requests and responses from multiple internal hosts to be mapped to a smaller number of external IP addresses, such as one address. Thus there could be many clients behind a NAT box that appear to come from the same source IP address to a server outside that NAT box. If one of these were an attacker (think Zombie or Botnet), that behind-NAT attacker could get the Server Cookie for some server for the outgoing IP address by just making some random request to that server. It could then include that Server Cookie in the COOKIE OPT of requests to the server with the forged local IP address of some other host and/or client behind the NAT box. (Attacker possession of this Server Cookie will not help in forging responses to cause cache poisoning as such responses are protected by the required Client Cookie.)

To fix this potential defect, it is necessary to distinguish different clients behind a NAT box from the point of view of the server. It is for this reason that the Server Cookie is specified as a pseudo-random function of both the request source IP address and the Client Cookie. From this inclusion of the Client Cookie in the calculation of the Server Cookie, it follows that a stable Client Cookie, for any particular server, is needed. If, for example, the request ID was included in the calculation of the Client Cookie, it

would normally change with each request to a particular server. This would mean that each request would have to be sent twice: first to learn the new Server Cookie based on this new Client Cookie based on the new ID and then again using this new Client Cookie to actually get an answer. Thus the input to the Client Cookie computation must be limited to the server IP address and one or more things that change slowly such as the client secret.

In principle, there could be a similar problem for servers, not due to NAT but due to mechanisms like anycast which may cause queries to a DNS server at an IP address to be delivered to any one of several machines. (External queries to a DNS server behind a NAT box usually occur via port forwarding such that all such queries go to one host.) However, it is impossible to solve this the way the similar problem was solved for NATed clients; if the Server Cookie was included in the calculation of the Client Cookie the same way the Client Cookie is included in the Server Cookie, you would just get an almost infinite series of errors as a request was repeatedly retried.

For servers accessed via anycast to successfully support DNS COOKIES, the server clones must either all use the same server secret or the mechanism that distributes queries to them must cause the queries from a particular client to go to a particular server for a sufficiently long period of time that extra queries due to changes in Server Cookie resulting from accessing different server machines are not unduly burdensome. (When such anycast-accessed servers act as recursive servers or otherwise act as clients they normally use a different unique address to source their queries to avoid confusion in the delivery of responses.)

For simplicity, it is RECOMMENDED that the same server secret be used by each DNS server in a set of anycast servers. If there is limited time skew in updating this secret in different anycast servers, this can be handled by a server accepting requests containing a Server Cookie based on either its old or new secret for the maximum likely time period of such time skew (see also Section 5.4).

7. Deployment

The DNS cookies mechanism is designed for incremental deployment and to complement the orthogonal techniques in [RFC5452]. Either or both techniques can be deployed independently at each DNS server and client.

In particular, a DNS server or client that implements the DNS COOKIE mechanism can interoperate successfully with a DNS client or server that does not implement this mechanism although, of course, in this case it will not get the benefit of the mechanism and the server involved might choose to severely rate limit responses. When such a server or client interoperates with a client or server which also implements the DNS cookies mechanism, they get the weak security benefits of the DNS Cookies mechanism.

8. IANA Considerations

IANA is requested to assign the following four code points:

The OPT option value for COOKIE is <TBD> [10 suggested].

Three new DNS error codes in the range above 16 and below 3,840 as shown below:

RCODE	Name	Description	Reference
-----	-----	-----	-----
TBD1[23]	NOCOKIE	No client cookie.	[this document]
TBD2[24]	MFCOKIE	Malformed cookie.	[this document]
TBD3[25]	BADCOKIE	Bad/missing server cookie.	[this document]

9. Security Considerations

DNS Cookies provide a weak form of authentication of DNS requests and responses. In particular, they provide no protection against "on-path" adversaries; that is, they provide no protection against any adversary that can observe the plain text DNS traffic, such as an on-path router, bridge, or any device on an on-path shared link (unless the DNS traffic in question on that path is encrypted).

For example, if a host is connected via an unsecured IEEE 802.11 link (Wi-Fi), any device in the vicinity that could receive and decode the 802.11 transmissions must be considered "on-path". On the other hand, in a similar situation but one where 802.11 Robust Security (WPAv2) is appropriately deployed on the Wi-Fi network nodes, only the Access Point via which the host is connecting is "on-path" as far as the 802.11 link is concerned.

Despite these limitations, deployment of DNS Cookies on the global Internet is expected to provide a substantial reduction in the available launch points for the traffic amplification and denial of service forgery attacks described in Section 2 above.

Should stronger message/transaction security be desired, it is suggested that TSIG or SIG(0) security be used (see Section 3.2); however, it may be useful to use DNS Cookies in conjunction with these features. In particular, DNS Cookies could screen out many DNS messages before the cryptographic computations of TSIG or SIG(0) are required and, if SIG(0) is in use, DNS Cookies could usefully screen out many requests given that SIG(0) does not screen requests but only authenticates the response of complete transactions.

9.1 Cookie Algorithm Considerations

The cookie computation algorithm for use in DNS Cookies SHOULD be based on a pseudo-random function at least as strong as [FNV] because an excessively weak or trivial algorithm could enable adversaries to guess cookies. However, in light of the weak plain-text token security provided by DNS Cookies, a strong cryptography hash algorithm may not be warranted in many cases, and would cause an increased computational burden. Nevertheless there is nothing wrong with using something stronger, for example, HMAC-SHA256-64 [RFC6234], assuming a DNS processor has adequate computational resources available. DNS processors that feel the need for somewhat stronger security without a significant increase in computational load should consider more frequent changes in their client and/or server secret; however, this does require more frequent generation of a cryptographically strong random number [RFC4086]. See Appendices A and B for specific examples of cookie computation algorithms.

Acknowledgements

The contributions of the following are gratefully acknowledged:

Tim Wicinski

Normative References

- [RFC2119] - Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4086] - Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC6891] - Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.

Informative References

- [FNV] - G. Fowler, L. C. Noll, K.-P. Vo, D. Eastlake, "The FNV Non-Cryptographic Hash Algorithm", draft-eastlake-fnv, work in progress.
- [RFC2845] - Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.
- [RFC2930] - Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.
- [RFC2931] - Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.
- [RFC3022] - Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC4033] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC4966] - Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.
- [RFC5452] - Hubert, A. and R. van Mook, "Measures for Making DNS More

Resilient against Forged Answers", RFC 5452, January 2009.

[RFC6234] - Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

Appendix A: Example Client Cookie Algorithms

A.1 A Simple Algorithm

An simple example method to compute Client Cookies is the FNV-64 [FNV] of the server IP address and the client secret. That is

$$\text{Client Cookie} = \text{FNV-64} (\text{Client Secret} \mid \text{Server IP Address})$$

where " \mid " indicates concatenation.

A.2 A More Complex Algorithm

A more complex algorithm to calculate Client Cookies is give below. It uses more computational resources than the simpler algorithm shown in A.1.

$$\text{Client Cookie} = \text{HMAC-SHA256-64} (\text{Client Secret}, \text{Server IP Address})$$

Appendix B: Example Server Cookie Algorithms

B.1 A Simple Algorithm

An example simple method producing a 64-bit Server Cookie is the FNV-64 [FNV] of the request IP address, the Client Cookie, and the server secret. That is

```
Server Cookie =  
    FNV-64 ( Server Secret | Request IP Address | Client Cookie )
```

where "|" represents concatenation.

B.2 A More Complex Algorithm

Since the Server Cookie is variable size, the server can store various information in that field as long as it is hard for an adversary to guess the entire quantity used for weak authentication. There should be 64 bits of entropy in the Server Cookie; for example it could have a sub-field of 64-bits computed pseudo-randomly with the server secret as one of the inputs to the pseudo-random function. Types of additional information that could be stored include a time stamp and/or a nonce.

The example below is one variation for the Server Cookie that has been implemented in a beta release of BIND where the Server Cookie is 128 bits composed as follows:

Sub-field	Size
-----	-----
Nonce	32 bits
Time	32 bits
Hash	64 bits

With this algorithm, the server sends a new 128-bit cookie back with every request. The Nonce field assures a low probability that there would be a duplicate.

The Time field gives the server time and makes it easy to reject old cookies.

The Hash part of the Server Cookie is the hard-to-guess part. In the beta release of BIND, its computation can be configured to use AES, HMAC-SHA1, or, as shown below, HMAC-SHA256:

```
hash =  
    HMAC-SHA256-64 ( Server Secret,  
        (Client Cookie | nonce | time | client IP Address) )  
where "|" represents concatenation.
```

Author's Address

Donald E. Eastlake 3rd
Huawei Technologies
155 Beaver Street
Milford, MA 01757 USA

Telephone: +1-508-333-2270
EMail: d3e3e3@gmail.com

Mark Andrews
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063 USA

Email: marka@isc.org

Copyright, Disclaimer, and Additional IPR Provisions

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 28, 2015

C. Grothoff
INRIA
M. Wachs
Technische Universitaet Muenchen
H. Wolf, Ed.
GNU consensus
J. Appelbaum
L. Ryge
Tor Project Inc.
January 24, 2015

Special-Use Domain Names of Peer-to-Peer Systems
draft-grothoff-iesg-special-use-p2p-names-04

Abstract

This document registers a set of Special-Use Domain Names for use with Peer-to-Peer (P2P) systems, as per RFC6761.

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 July 28, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Applicability	3
3. Terminology and Conventions Used in This Document	4
4. Description of Special-Use Domains in P2P Networks	5
4.1. The "GNU" Relative pTLD	5
4.2. The "ZKEY" Compressed Public Key pTLD	6
4.3. Geographically Anonymous pTLDs	8
4.3.1. The "ONION" Hidden Service pTLD	8
4.3.2. The "EXIT" Client Source Routing pTLD	10
4.3.3. The "I2P" Addressbook pTLD	12
4.4. The "BIT" Timeline System pTLD	14
5. Security Considerations	16
6. IANA Considerations	18
7. Acknowledgements	18
8. References	19
8.1. Normative References	19
8.2. Informative References	19
Authors' Addresses	21

1. Introduction

The Domain Name System (DNS) is primarily used to map human-memorable names to IP addresses, which are used for routing but generally not meaningful for humans.

Peer-to-Peer (P2P) systems use specific decentralized mechanisms to allocate, register, manage, and resolve names. However, the hierarchical nature of DNS makes it unsuitable for various P2P Name Systems. Such P2P Name Systems operate entirely outside of DNS, independently from the DNS root and delegation tree.

As compatibility with applications using domain names is desired, these P2P overlay networks often define exclusive alternative Top-Level Domains to avoid conflict between the P2P namespace and the DNS hierarchy.

In order to avoid interoperability issues with DNS as well as to address security and privacy concerns, this document registers a set of Special-Use Domain Names for use with P2P systems (pTLDs), as per [RFC6761],: "GNU", "ZKEY", "ONION", "EXIT", "I2P", and "BIT".

The GNU Name System (GNS) ("GNU", "ZKEY"), the Tor network ("ONION", "EXIT"), the Invisible Internet Project ("I2P"), and the Dot-Bit Project ("BIT") use these pTLDs to realize fully-decentralized and censorship-resistant naming. The "EXIT" pTLD is used to control overlay routing and to securely specify path selection choices [TOR-PATH].

2. Applicability

[RFC6761] Section 3 states:

"[I]f a domain name has special properties that affect the way hardware and software implementations handle the name, that apply universally regardless of what network the implementation may be connected to, then that domain name may be a candidate for having the IETF declare it to be a Special-Use Domain Name and specify what special treatment implementations should give to that name. On the other hand, if declaring a given name to be special would result in no change to any implementations, then that suggests that the name may not be special in any material way, and it may be more appropriate to use the existing DNS mechanisms [RFC1034] to provide the desired delegation, data, or lack-of-data, for the name in question. Where the desired behaviour can be achieved via the existing domain name registration processes, that process should be used. Reservation of a Special-Use Domain Name is not a mechanism for circumventing normal domain name registration processes."

The set of Special-Use Domain Names for Peer-to-Peer Systems (pTLDs) reserved by this document meet this requirement, as they share the following specificities:

- o pTLDs are not manageable by some designated administration. Instead, they are managed according to various alternate strategies or combinations thereof, introduced in this document, and their respective protocol specifications: automated cryptographic assignment (".onion", ".zkey"), user-controlled assignment in a private scope (".gnu", ".i2p"), or in a global public ledger (".bit"), or used as a source-routing mechanism to delegate DNS resolution to a remote peer (".exit").
- o pTLDs do not depend on the DNS context for their resolution: GNS and Namecoin domains MAY use the DNS servers infrastructure, as they return DNS-compatible results; and all pTLDs use specific P2P protocols for regular name resolution, covered by their respective protocol specifications.

- o When a pTLD protocol has been implemented, the implementation MUST intercept queries for the pTLD to ensure P2P names cannot leak into the DNS.
- o The appropriate pTLD protocols can be implemented in existing software libraries and APIs to extend regular DNS operation and enable P2P name resolution. However, the default hierarchical DNS response to any request to any pTLD MUST be NXDOMAIN.
- o Finally, in order for pTLDs to realize a censorship-resistant, fully-decentralized name system, and provide security and privacy features matching user expectations, this document specifies changes required in existing DNS software and DNS operations.

3. Terminology and Conventions Used in This Document

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 [RFC2119].

The word "peer" is used in the meaning of a individual system on the network.

The abbreviation "pTLD" is used in this document to mean a pseudo Top-Level Domain, i.e., a Special-Use Domain Name per [RFC6761] reserved to P2P Systems in this document. A pTLD is mentioned in capitals, and within double quotes to mark the difference with a regular DNS gTLD.

In this document, ".tld" (lowercase, with quotes) means: any domain or hostname within the scope of a given pTLD, while .tld (lowercase, without quotes) refers to an adjective form. For example, a collection of ".gnu" peers in "GNU", but an .onion URL. [TO REMOVE: in the IANA Considerations section, we use the simple .tld format to request TLD reservation for consistency with previous RFCs].

The word "NXDOMAIN" refers to an alternate expression for the "Name Error" RCODE as described in section 4.1.1 of [RFC1035]. When referring to "NXDOMAIN" and negative caching [RFC2308] response, this document means an authoritative (AA=1) name error (RCODE=3) response exclusively.

The Tor-related names such as 'circuit', 'exit', 'node', 'relay', 'stream', and related Tor terms are described in [Dingledine2004] and the Tor protocol specification [TOR-PROTOCOL].

The I2P-related names such as 'Destination' are described in [zzz2009].

4. Description of Special-Use Domains in P2P Networks

4.1. The "GNU" Relative pTLD

"GNU" is used to specify that a domain name should be resolved using GNS. The GNS resolution process is documented in [Wachs2014].

The "GNU" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

Since there is no central authority responsible for assigning .gnu names, and that specific domain is local to the local peer, users need to be aware of that specificity.

Legacy applications MAY expect the DNS-to-GNS proxy to return DNS compatible results for the resolution of .gnu domains.

2. Legacy application software does not need to recognize .gnu domains as special, and may continue to use these names as they would other domain names.

GNS-aware applications MAY also use GNS resolvers directly to resolve .gnu domains (in particular, if they want access to GNS-specific record types).

3. Name resolution APIs and libraries SHOULD either respond to requests for .gnu names by resolving them via the GNS protocol, or respond with NXDOMAIN.

4. Caching DNS servers SHOULD recognize .gnu names as special and SHOULD NOT attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .gnu names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.

5. Authoritative DNS servers are not expected to treat .gnu domain requests specially. In practice, they MUST answer with NXDOMAIN,

as "GNU" is not available via global DNS resolution, and not doing so can put users' privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .gnu names are reserved for use with GNS, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).
7. DNS registries/registrar MUST NOT grant any request to register .gnu names. This helps avoid conflicts [SAC45]. These names are defined by the GNS protocol specification, and they fall outside the set of names available for allocation by registries/registrar.

4.2. The "ZKEY" Compressed Public Key pTLD

The "ZKEY" pTLD is used to signify that resolution of the given name MUST be performed using a record signed by an authority that is in possession of a particular public key. Names in "ZKEY" MUST end with a domain which is the compressed point representation from [EdDSA] on [Curve25519] of the public key of the authority, encoded using Crockford's variant of base32hex [RFC4648] (with additionally 'U' being considered equal to 'V') for easier optical character recognition. A GNS resolver uses the key to locate a record signed by the respective authority.

"ZKEY" provides a (reverse) mapping from globally unique hashes to public key, therefore .zkey names are non-memorable, and are expected to be hidden from the user [Wachs2014].

The "ZKEY" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

Since there is no central authority necessary or possible for assigning .zkey names, and those names match cryptographic keys, users need to be aware that they do not belong to regular DNS, but are still global in their scope.

Legacy applications MAY expect the DNS-to-GNS proxy to return DNS-compatible results for the resolution of .zkey domains.

2. Application software does not need to recognize .zkey domains as special, and may continue to use these names as they would other domain names.

GNS-aware applications MAY also use GNS resolvers directly to resolve .zkey domains

3. Name resolution APIs and libraries SHOULD either respond to requests for .zkey names by resolving them via the GNS protocol, or respond with NXDOMAIN.
4. Caching DNS servers SHOULD recognize .zkey names as special and SHOULD NOT attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .zkey names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.
5. Authoritative DNS Servers are not expected to treat .zkey domain requests specially. In practice, they MUST answer with NXDOMAIN, as "ZKEY" is not available via global DNS resolution, and not doing so MAY put users' privacy at risk (see item 6).
6. DNS server operators SHOULD be aware that .zkey names are reserved for use with GNS, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).
7. DNS registries/registrars MUST NOT grant any request to register .zkey names. This helps avoid conflicts [SAC45]. These names are defined as described above, and they fall outside the set of names available for allocation by registries/registrars.

4.3. Geographically Anonymous pTLDs

Both the Tor "Onionspace" and the I2P network are designed to provide geographic anonymity to services and all clients visiting them. They provide additional properties such as NAT traversal, strong authentication, anonymity, and censorship resistance.

The Tor anonymization network makes use of several special pTLD labels, three of which have seen widespread usage to date. This document introduces two of them, "ONION" and "EXIT". The interested reader is invited to refer to [TOR-ADDRESS] for further information on the "NOCONNECT" pTLD, whose limited testing scope does not warrant the attention of the larger Internet community.

The I2P network uses a single pTLD, "I2P", but the specific subdomain "B32.I2P" offers properties similar to Tor's "ONION" and GNS's "ZKEY"

The public literature often uses the term "Hidden Service" to refer to both Tor's Hidden Service protocol and services, and I2P's Destinations. This term suggests that such services are hidden from view, whereas only their geographic location is unknown: given their name and the appropriate name resolver, such services are as much accessible as any other regular Web site or Internet service.

4.3.1. The "ONION" Hidden Service pTLD

The widely deployed "ONION" designates the "Onionspace", an anonymous Tor Hidden Service reachable via the Tor network [Dingledine2004]. These .onion hostnames are self-authenticating addresses for use with any TCP service.

Addresses in "ONION" are opaque, non-mnemonic, alpha-semi-numeric digest hashes corresponding to the unique identity key of a given Tor hidden service. Therefore such .onion addresses are self-authenticating. The algorithm to obtain the .onion hash from the Tor hidden service's public key is out of scope of this document, and described in the Tor Address specification [TOR-ADDRESS]. Tor generates this "Onion key" automatically when the hidden service is configured. Tor clients use it following the Tor Rendezvous specifications [TOR-RENDEZVOUS].

The "ONION" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

Since there is no central authority necessary or possible for assigning .onion names, and those names correspond to cryptographic keys, users need to be aware that they do not belong to regular DNS, but are still global in their scope.

2. Application software MAY recognize .onion domains as special, and SHOULD use these names as they would other domain names.

Application software MAY implement mechanisms helping the user to ensure their privacy expectations are met, such as warning the user if they do not detect an active local Tor resolver, displaying a warning on first-use of an .onion domain to explain the necessity of a Tor resolver to reach such domains, etc.

If an application knows how to differentiate between DNS and P2P name resolution, it:

- * SHOULD NOT pass requests for .onion domains to DNS resolvers or libraries,
- * MUST expect NXDOMAIN as the only valid DNS response, and
- * SHOULD treat other answers from DNS as errors.

Tor-aware applications MAY also use Tor resolvers directly.

3. Name resolution APIs and libraries SHOULD either respond to requests for .onion names by resolving them via the Tor protocol, or respond with NXDOMAIN.
4. Caching DNS servers SHOULD recognize .onion names as special and SHOULD NOT attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .onion names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.
5. Authoritative DNS servers are not expected to treat .onion domain requests specially. In practice, they MUST answer with NXDOMAIN, as "ONION" is not available via global DNS resolution, and not doing so MAY put users' privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .onion names are reserved for use with Tor, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).
7. DNS registries/registrars MUST NOT grant any request to register .onion names. This helps avoid conflicts [SAC45]. These names are defined the Tor protocol specification [TOR-PROTOCOL], and they fall outside the set of names available for allocation by registries/registrars.

4.3.2. The "EXIT" Client Source Routing pTLD

The .exit suffix is used as an in-band source routing control channel, usually for selection of a specific Tor relay during path creation as the last node in the Tor circuit.

It may be used to access a DNS host via specific Torservers, in the form "hostname.nickname-or-fingerprint.exit", where the "hostname" is a valid hostname, and the "nickname-or-fingerprint" is either the nickname of a Tor relay in the Tor network consensus, or the hex-encoded SHA1 digest of the given node's public key (fingerprint).

For example, "gnu.org.noisetor.exit" will route the client to "gnu.org" via the Tor node nicknamed "noisetor". Using the fingerprint instead of the nickname ensures that the path selection uses a specific Tor exit node, and is harder to remember: e.g., "gnu.org.f97f3b153fed6604230cd497a3d1e9815b007637.exit".

When Tor sees an address in this format, it uses the specified "nickname-or-fingerprint" as the exit node. If no "hostname" component is given, Tor defaults to the published IPv4 address of the Tor exit node [TOR-EXTSOCKS].

Because "hostname" is allegedly valid, the total length of a .exit construct may exceed the maximum length allowed for domain names. Moreover, the resolution of "hostname" happens at the exit node. Trying to resolve such invalid domain names, including chaining .exit names will likely return a DNS lookup failure at the first exit node.

The "EXIT" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

Since .exit names correspond to a Tor-specific routing construct to reach target hosts via chosen Tor exit nodes, users need to be aware that they do not belong to regular DNS and that the actual target precedes the second-level domain name.

2. Application software MAY recognize that .exit domains are special and when they do SHOULD NOT pass requests for these domains to DNS resolvers and libraries.

As mentioned in items 4 and 5 below, regular DNS resolution is expected to respond with NXDOMAIN. Therefore, if it can differentiate between DNS and P2P name resolution, application software:

- * MUST expect NXDOMAIN as the only valid DNS response, and
- * SHOULD treat other answers from DNS as errors.

Tor-aware applications MAY also use Tor resolvers directly.

3. Name resolution APIs and libraries SHOULD either respond to requests for .exit names by resolving them via the Tor protocol, or respond with NXDOMAIN.
4. Caching DNS servers SHOULD recognize .exit names as special and SHOULD NOT, by default, attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .exit names. Instead, caching DNS servers SHOULD, by default, generate immediate negative responses for all such queries.
5. Authoritative DNS servers are not expected to treat .exit domain requests specially. In practice, they MUST answer with NXDOMAIN, as "EXIT" is not available via global DNS resolution, and not doing so MAY put users' privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .exit names are reserved for use with Tor, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).
7. DNS registries/registrars MUST NOT grant any request to register .exit names. This helps avoid conflicts [SAC45]. These names are defined by the Tor address specification, and they fall outside the set of names available for allocation by registries/registrars.

4.3.3. The "I2P" Addressbook pTLD

"I2P" provides accessibility to hidden services within the I2P network [zzz2009]. I2P is a scalable, self-organizing, resilient packet switched anonymous network layer, upon which any number of different anonymity or security-conscious applications can operate, using any protocol.

I2P hidden services and clients are identified by Destinations, anonymous analogues of IP addresses. The "I2P" pTLD, chosen in 2003 [I2P-CHOICE], houses two methods for looking up Destinations:

A local table called the addressbook stores a map of .i2p addresses to Destinations. Each user maintains their own mappings that can be shared with others, allowing them to "discover" new names by importing published addressbooks of peers, and they can emulate traditional DNS by choosing to treat these peers as name servers. The comparison however stops here, as only local uniqueness is mandated. As the system is decentralized, "example.i2p" may resolve differently for different peers depending on the state of their respective addressbooks.

To address globally unique names, the I2P developers dedicated the "B32.I2P" subdomain to hold Base32-encoded [RFC4648] references to Destinations. Like .onion addresses, .b32.i2p addresses are self-authenticating. The details of the encoding are out of scope for this document, and documented in [I2P-NAMING]. The purpose of .b32.i2p addresses is similar to ".zkey", that is to enable (reverse) mapping for a globally unique hidden service that may not have a defined entry in the local addressbook.

The "I2P" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

Since there is no central authority responsible for assigning .i2p names, and that the ultimate mapping is decided by the local peer, users need to be aware of that specificity.

2. Application software SHOULD recognize .i2p domains as special and SHOULD NOT use them as they would other domains.

Applications SHOULD NOT pass requests for .i2p domains to DNS resolvers and libraries.

As mentioned in points 4 and 5 below, regular DNS resolution is expected to respond with NXDOMAIN. Therefore, if it can differentiate between DNS and P2P name resolution, application software can expect such a response, and can choose to treat other responses from resolvers and libraries as errors.

3. Name resolution APIs and libraries SHOULD either respond to requests for .i2p names by resolving them via the I2P protocol, or respond with NXDOMAIN.
4. Caching DNS servers SHOULD recognize .i2p names as special and SHOULD NOT attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .i2p names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.
5. Authoritative DNS servers are not expected to treat .i2p domain requests specially. In practice, they MUST answer with NXDOMAIN, as "I2P" is not available via global DNS resolution, and not doing so MAY put users' privacy at risk (see item 6).
6. DNS server operators SHOULD be aware that .i2p names are reserved for use with I2P, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .i2p names. This helps avoid conflicts [SAC45]. These names are defined by the I2P protocol specification, and they fall outside the set of names available for allocation by registries/registrars.

4.4. The "BIT" Timeline System pTLD

Namecoin is a timeline-based system in the style of Bitcoin to create a global, secure, and memorable name system. It creates a single, globally accessible, append-only timeline of name registrations. Timeline-based systems rely on a peer-to-peer network to manage updates and store the timeline. In the Namecoin system, modifications to key-value mapping are attached to transactions which are committed to the timeline by "mining". Mining is the use of brute-force methods to find (partial) hash collisions with a state summary (fingerprint) representing the complete global state -- including the full history -- of the timeline.

"BIT" provides a name space where names are registered via transactions in the Namecoin currency [Namecoin]. Like Bitcoins, Namecoins are created using a proof-of-work calculation, which is also used to establish a decentralized, multi-party consensus on the valid transaction history, and thus the set of registered names and their values [SquareZooko].

The Namecoin used in a transaction to register a name in "BIT" is lost. This is not a fundamental problem as more coins can be generated via mining (proof-of-work calculations). The registration cost is set to decrease over time, to prevent early adopters from registering too many names.

The owner of a name can update the associated value by issuing an update, which is a transaction that uses a special coin. This coin is generated as change during the registration operation. If a name is not updated for a long time, the registration expires.

Performing a lookup for a name with Namecoin consists in checking the timeline for correctness to ensure the validity of the blockchain, and traversing it to see if it contains an entry for the desired name. Namecoin supports resolution for other peer-to-peer systems such as ".onion" and ".i2p" via specific resource records.

Like DNS registry, the Dot-Bit registry is public. But unlike DNS, the public registry is maintained by network consensus on the blockchain. It departs from DNS in three ways:

first, domain names are not delegated to an authority that can assign them, but acquired by the operating party (the "domain owner"), in the form of a historical claim made directly by appending to the Namecoin blockchain. The domain is thus bound not to a legal contract with an administrative authority, but to a cryptographic coin, and the network consensus on the timeline.

second, the timeline contains the entire registry for all .bit domains: the Namecoin blockchain itself is the complete domain database. As participant peers maintain the consensus on the timeline, they store a local copy of the Namecoin blockchain. Therefore, to those peers, name resolution and registry traversal are both local and private. Each participant theoretically owns the whole domain's database. In practice, some users can trust a name server to access the Namecoin blockchain on their behalf.

third, the Namecoin system is not limited to domain names and can store arbitrary data types. Each record must follow the same rules (expiry time, data size limits, etc.). The Namecoin's Domain Name Specification [Namecoin-DNS] defines the "d namespace" for use with "BIT" and other unrelated namespaces co-exist on the Namecoin blockchain.

The "BIT" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

From the user's perspective, the resolution of .bit names is similar to the normal DNS resolution, and thus should not affect normal usage of most Internet applications.

2. Application software SHOULD NOT recognize .bit domains as special and SHOULD treat them as they would other domains.

Applications MAY pass requests to the "BIT" pTLD to DNS resolvers and libraries if A/AAAA records are desired. If available, the local resolver can intercept such requests within the respective operating system hooks and return DNS-compatible results.

Namecoin-aware applications MAY choose to talk directly to the respective P2P resolver, and use this to access additional record types that are not defined in DNS.

3. Name resolution APIs and libraries SHOULD either respond to requests for .bit names by resolving them via the Namecoin protocol, or respond with NXDOMAIN.
4. Caching DNS servers SHOULD recognize .bit names as special and SHOULD NOT attempt to resolve them. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.

Given that .bit users typically have no special privacy expectations, and those names are globally unique, local caching DNS servers MAY choose to treat them as regular domain names, and cache the responses obtained from the Namecoin blockchain. In that case however, NXDOMAIN results SHOULD NOT be cached, as new .bit domains may become active at any time.

5. Authoritative DNS servers are not expected to treat .bit domain requests specially. In practice, they MUST answer with NXDOMAIN, as "BIT" is not available via global DNS resolution.
6. DNS server operators SHOULD be aware that .bit names are reserved for use with Namecoin, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).
7. DNS registries/registrars MUST NOT grant any request to register .bit names. This helps avoid conflicts [SAC45]. These names are defined by the Namecoin protocol specification, and they fall outside the set of names available for allocation by registries/registrars.

5. Security Considerations

Specific software performs the resolution of the six Special-Use Domain Names presented in this document; this resolution process happens outside of the scope of DNS. Leakage of requests to such domains to the global operational DNS can cause interception of traffic that might be misused to monitor, censor, or abuse the user's

trust, and lead to privacy issues with potentially tragic consequences for the user.

This document reserves these Top-Level Domain names to minimize the possibility of confusion, conflict, and especially privacy risks for users.

In the introduction of this document, there's a requirement that DNS operators do not override resolution of the P2P Names. This is a regulatory measure and cannot prevent such malicious abuse in practice. Its purpose is to limit any information leak that would result from incorrectly configured systems, and to avoid that resolvers make unnecessary contact to the DNS Root Zone for such domains. Verisign, Inc., as well as several Internet service providers (ISPs) have notoriously abused their position to override NXDOMAIN responses to their customers in the past. For example, if a DNS operator would decide to override NXDOMAIN and send advertising to leaked .onion sites, the information leak to the DNS would extend to the advertising server, with unpredictable consequences. Thus, implementors should be aware that any positive response coming from DNS must be considered with extra care, as it suggests a leak to DNS has been made, contrary to user's privacy expectations.

The reality of X.509 Certificate Authorities (CAs) creating misleading certificates for these pTLDs due to ignorance stresses the need to document their special use. X.509 Certificate Authorities MAY create certificates for "ONION", "BIT", and "ZKEY" given CSRs signed with the respective private keys corresponding to the respective names. For "BIT", the Certificate Authority SHOULD limit the expiration time of the certificate to match the registration. Certificate Authorities MUST NOT create certificates for the "EXIT", "GNU", and "I2P" Top-Level domains. Nevertheless, clients SHOULD accept certificates for these Top-Level domains as they may be created legitimately by local proxies on the fly.

[SAC57] reports, page 11, that the CA/Browser forum stated: "Also as of the Effective Date [1 July 2012], the CA SHALL NOT issue a certificate with an Expiry Date later than 1 November 2015 with a subjectAlternativeName extension or Subject commonName field containing a Reserved IP Address or Internal Server Name."

It is not clear whether e.g., .onion sites are considered "Internal Server Names", however, we can expect that services doubling their public Web site with an onion site would use a single SSL certificate for both, as did Facebook with "facebookcorewwi.onion". Given this forum also declared the CAs would revoke such SSL certificates in October 2016, that opens a three (now two) years period of vulnerability for new gTLDs to suffer MiTM attacks over HTTPS. Such

practice by CAs to validate certificates to invalid TLDs without verification may lead, e.g., to malicious third parties without any relation to an existing .onion site to register a fake certificate for that site in order to facilitate attacks, especially when combined with name collision risk as explained in [SAC62].

Because the Namecoin system uses a timeline-based blockchain for name assignment and resolution, it grants query privacy to the users who maintain their own copy of the blockchain (Section 4.4), but the entire zone of a .bit domains is publicly available in the Namecoin blockchain, making enumeration of names within a .bit zone ("zone walking") a trivial attack to conduct. This might be a concern to some domain operators as it exposes their infrastructure to potential adversaries. That concern may be addressed in future versions of Namecoin, but the records already in the blockchain will remain there unprotected.

Finally, legacy applications that do not explicitly support the pTLDs significantly increase the risk of pTLD queries escaping to DNS, as they are entirely dependent on the correct configuration on the operating system.

6. IANA Considerations

The Internet Assigned Numbers Authority (IANA) reserved the following entries in the Special-Use Domain Names registry [RFC6761]:

.gnu
.zkey
.onion
.exit
.i2p
.bit

[TO REMOVE: the assignment URL is <https://www.iana.org/assignments/special-use-domain-names/>]

7. Acknowledgements

The authors thank the I2P and Namecoin developers for their constructive feedback, as well as Mark Nottingham for his proof-reading and valuable feedback. The authors also thank the members of DNSOP WG for their critiques and suggestions.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

8.2. Informative References

- [Curve25519] Bernstein, D., "Curve25519: new Diffie-Hellman speed record", February 2006, <<http://cr.yp.to/ecdh/curve25519-20060209.pdf>>.
- [Dingledine2004] Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004, <<https://www.onion-router.net/Publications/tor-design.pdf>>.
- [EdDSA] Bernstein, D., Duif, N., Lange, T., Schwabe, P., and Y. Yang, "High-speed, high-security signatures", September 2011, <<http://ed25519.cr.yp.to/ed25519-20110926.pdf>>.
- [I2P-CHOICE] Hacker, J. and The I2P Community, "I2P Dev Meeting 059", September 2003, <<https://geti2p.net/en/meetings/059>>.
- [I2P-NAMING] Hacker, J. and The I2P Community, "Naming in I2P and Addressbook", April 2014, <<https://geti2p.net/en/docs/naming>>.

- [Namecoin]
The .bit Project, "Namecoin", 2013,
<<https://namecoin.org/>>.
- [Namecoin-DNS]
The .bit Project, "Namecoin Domain Name Specification",
2015, <<https://bit.namecoin.org/spec>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
Encodings", RFC 4648, October 2006.
- [SAC45] ICANN Security and Stability Advisory Committee, "Invalid
Top Level Domain Queries at the Root Level of the Domain
Name System", November 2010, <[http://www.icann.org/en/
groups/ssac/documents/sac-045-en.pdf](http://www.icann.org/en/groups/ssac/documents/sac-045-en.pdf)>.
- [SAC57] ICANN Security and Stability Advisory Committee, "SSAC
Advisory on Internal Name Certificates", March 2013,
<[http://www.icann.org/en/groups/ssac/documents/
sac-057-en.pdf](http://www.icann.org/en/groups/ssac/documents/sac-057-en.pdf)>.
- [SAC62] ICANN Security and Stability Advisory Committee, "SSAC
Advisory Concerning the Mitigation of Name Collision
Risk", November 2013, <[http://www.icann.org/en/groups/
ssac/documents/sac-062-en.pdf](http://www.icann.org/en/groups/ssac/documents/sac-062-en.pdf)>.
- [SquareZooko]
Swartz, A., "Squaring the Triangle: Secure, Decentralized,
Human-Readable Names", 2011,
<<http://www.aaronsw.com/weblog/squarezooko>>.
- [TOR-ADDRESS]
Mathewson, N. and R. Dingledine, "Special Hostnames in
Tor", September 2011, <[https://gitweb.torproject.org/
torspec.git/plain/address-spec.txt](https://gitweb.torproject.org/torspec.git/plain/address-spec.txt)>.
- [TOR-EXTSOCKS]
Mathewson, N. and R. Dingledine, "Tor's extensions to the
SOCKS protocol", February 2014, <[https://gitweb.torproject
.org/torspec.git/plain/socks-extensions.txt](https://gitweb.torproject.org/torspec.git/plain/socks-extensions.txt)>.
- [TOR-PATH]
Mathewson, N. and R. Dingledine, "Tor Path Specification",
November 2014, <[https://gitweb.torproject.org/torspec.git/
plain/path-spec.txt](https://gitweb.torproject.org/torspec.git/plain/path-spec.txt)>.

[TOR-PROTOCOL]

Dingledine, R. and N. Mathewson, "Tor Protocol Specification", August 2014, <<https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt>>.

[TOR-RENDEZVOUS]

Mathewson, N. and R. Dingledine, "Tor Rendezvous Specification", April 2014, <<https://gitweb.torproject.org/torspec.git/plain/rend-spec.txt>>.

[Wachs2014]

Wachs, M., Schanzenbach, M., and C. Grothoff, "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System", October 2014, <<https://gnunet.org/gns-paper>>.

[zzz2009] The I2P Project and L. Schimmer, "Peer Profiling and Selection in the I2P Anonymous Network", January 2009, <https://geti2p.net/_static/pdf/I2P-PET-CON-2009.1.pdf>.

Authors' Addresses

Christian Grothoff
INRIA
Equipe Decentralisee
INRIA Rennes Bretagne Atlantique
263 avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes, Bretagne F-35042
FR

Email: christian@grothoff.org

Matthias Wachs
Technische Universitaet Muenchen
Free Secure Network Systems Group
Lehrstuhl fuer Netzarchitekturen und Netzdienste
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: wachs@net.in.tum.de

Hellekin O. Wolf (editor)
GNU consensus

Email: hellekin@gnu.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Leif Ryge
Tor Project Inc.

Email: leif@synthesize.us

DNSOP
Internet-Draft
Intended status: Best Current Practice
Expires: July 20, 2014

W. Hardaker
Parsons
O. Gudmundsson
Shinkuro Inc.
S. Krishnaswamy
Parsons
January 16, 2014

DNSSEC Roadblock Avoidance
draft-hardaker-dnsop-dnssec-roadblock-avoidance-01.txt

Abstract

This document describes problems that a DNSSEC aware resolver/application might run into within a non-compliant infrastructure. It outline potential detection and mitigation techniques. The scope of the document is to create a shared approach to detect and overcome network issues that a DNSSEC software/system may face.

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 July 20, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Background	3
1.2. Notation	3
2. Goals	4
3. Detecting DNSSEC Non-Compliance	4
3.1. Determining DNSSEC support in neighboring recursive resolvers	4
3.1.1. Supports UDP answers	4
3.1.2. Supports TCP answers	5
3.1.3. Supports EDNS0	5
3.1.4. Supports the DO bit	5
3.1.5. Supports the AD bit	6
3.1.6. Returns RRsigs for signed answer	6
3.1.7. Supports querying for DNSKEY records	6
3.1.8. Supports querying for DS records	7
3.1.9. Supports negative answers with NSEC records	7
3.1.10. Supports negative answers with NSEC3 records	7
3.1.11. Supports queries where DNAME records lead to an answer	8
3.1.12. Permissive DNSSEC	8
3.1.13. UDP size limits	8
3.1.14. Supports Unknown RRtypes	8
3.2. Direct Network Queries	9
3.2.1. Support for Remote UDP Over Port 53	9
3.2.2. Support for Remote UDP With Fragmentation	9
3.2.3. Support for Outbound TCP Over Port 53	10
4. Aggregating The Results	10
4.1. Resolver capability description	10
5. Roadblock Avoidance	11
5.1. Partial Resolver Usage	13
5.1.1. Known Insecure Lookups	13
5.1.2. Partial NSEC/NSEC3 Support	14
6. Start-Up and Network Connectivity Issues	14
6.1. What To Do	14
7. IANA Considerations	15
8. Security Considerations	15
9. Acknowledgments	15
10. Normative References	15
Authors' Addresses	15

1. Introduction

This document describes problems with DNSSEC ([RFC4034], [RFC4035]) deployment due to non-compliant infrastructure. It poses potential detection and mitigation techniques.

1.1. Background

Deployment of DNSSEC today is hampered by network components that make it difficult or sometimes impossible for validating resolvers to effectively obtain the DNSSEC data they need. This can occur for many different reasons including

- o Because neighboring recursive resolvers are not fully DNSSEC compliant
- o Because resolvers are not even DNSSEC aware
- o Because "middle-boxes" active block/restrict outbound traffic to the DNS port (53) either UDP and/or TCP .
- o Network component in path does not allow UDP fragments
- o etc...

This document talks about ways a Host Validator can detect the state of the network it is attached to, and ways to hopefully circumvent the problems associated with the network defects it discovers. The tests described in this document may be performed on any validating resolver to detect and prevent problems. While these recommendations are mainly aimed at Host Validators it is prudent to perform these tests from regular Validating Resolvers before enabling just to make sure things work.

1.2. Notation

When we talk about a "Host Validator", this can either be a library that an application has linked in or an actual validating resolver running on the same machine.

A variant of this is a "Validating Forwarding Resolver", which is a resolver that is configured to use upstream Resolvers if possible. Validating Forward Resolver needs to perform the same set of tests before using an upstream recursive resolver.

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 [RFC2119].

2. Goals

The result of this work is intended to show how a Host Validator can detect the capabilities of a nearby recursive resolver, and work around any problems that could potentially affect DNSSEC resolution. This enables the Host Validator to make use of the caching functionality of the recursive resolver, which is desirable in that it decreases network traffic and improves response times.

A Host Validator has two choices: it can wait to determine that it has problems with a recursive resolver based on the results that it is getting from real-world queries issued to it, or it can proactively test for problems (Section Section 3) to build a work around list ahead of time (Section Section 5). There are pros and cons to both of these paths that are application specific, and this document does not attempt to provide guidance about whether proactive tests should or should not be used. Either way, DNSSEC roadblock avoidance techniques ought to be used when needed and if possible.

3. Detecting DNSSEC Non-Compliance

A Host Validator may choose to determine early-on what bottlenecks exist that may hamper its ability to perform DNSSEC look-ups. This section outlines tests that can be done to test certain features of the surrounding network.

NOTE: when performing these tests against an address, we make the following assumption about that address: It is a unicast address or an anycast cluster where all servers have identical configuration and connectivity.

3.1. Determining DNSSEC support in neighboring recursive resolvers

Ideally, a Host Validator can make use of the caching present in neighboring recursive resolvers. This section discusses the tests that a neighboring recursive resolver MUST pass in order to be fully usable as a near-by DNS cache.

Unless stated otherwise, all of the following tests SHOULD have the recursive flag set when sending out a query and SHOULD be sent over UDP. Unless otherwise stated, the tests MUST NOT have the DO bit set or utilize any of the other DNSSEC related requirements, like EDNS0. The tests are designed to check for one feature at a time.

3.1.1. Supports UDP answers

Purpose: This tests basic DNS over UDP functionality to a resolver.

Test: A DNS request is sent to the resolver under test for an A record for a known existing domain, such as `www.dnssec-tools.org`.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

Note: an implementation MAY chose to not perform the rest of the tests if this test fails, as clearly the resolver under test is severely broken.

3.1.2. Supports TCP answers

Purpose: This tests basic TCP functionality to a resolver.

Test: A DNS request is sent over TCP to the resolver under test for an A record for a known existing domain, such as `www.dnssec-tools.org`.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

3.1.3. Supports EDNS0

Purpose: Test whether a resolver properly supports the EDNS0 extension option.

Pre-requisite: "Supports UDP or TCP".

Test: Send a request to the resolver under test for an A record for a known existing domain, such as `www.dnssec-tools.org`, with an EDNS0 OPT record in the additional section.

SUCCESS: A DNS response was received that contains an EDNS0 option with version number 0.

3.1.4. Supports the DO bit

Purpose: This tests whether a resolver has minimal support of the DO bit.

Pre-requisite: "Supports EDNS0".

Test: Send a request to the resolver under test for an A record for a known existing domain such as `www.dnssec-tools.org`. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains the DO bit set.

Note: this only tests that the resolver sets the DO bit in the response. Later checks will determine if the DO bit was actually made use of. Some resolvers successfully pass this test because they simply copy the unknown flags into the response. Don't worry, they'll fail the later tests.

3.1.5. Supports the AD bit

Purpose: This tests whether the resolver is a validating resolver.

Pre-requisite: "Supports the DO bit".

Test: Send a request to the resolver under test for an A record for a known existing domain in a DNSSEC signed zone which is verifiable to a configured trust anchor, such as www.dnssec-tools.org using the root's published DNSKEY or DS record as a trust anchor. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains the AD bit set.

3.1.6. Returns RRSig for signed answer

Purpose: This tests whether a resolver will properly return RRSIG records when the DO bit is set.

Pre-requisite: "Supports the DO bit".

Test: Send a request to the resolver under test for an A record for a known existing domain in a DNSSEC signed zone, such as www.dnssec-tools.org. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains at least one RRSIG record.

3.1.7. Supports querying for DNSKEY records

Purpose: This tests whether a resolver can query for and receive a DNSKEY record from a signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an DNSKEY record which is known to exist in a signed zone, such as dnssec-tools.org/DNSKEY. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DNSKEY record in the answer section.

Note: Some DNSKEY RRset's are large and if the network path has problems with large answers this query may result in either false positive or false negative. In general the DNSKEY queried for is a small enough to fit into 1220 byte answer, to avoid false negative result when TCP is disabled. However, querying many zones will result in answers greater than 1220 bytes so ideally TCP MUST be available.

3.1.8. Supports querying for DS records

Purpose: This tests whether a resolver can query for and receive a DS record from a signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an DS record which is known to exist in a signed zone, such as `dnssec-tools.org/DS`. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DS record in the answer section.

3.1.9. Supports negative answers with NSEC records

Purpose: This tests whether a resolver properly returns NSEC records for a non-existing domain in a DNSSEC signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an A record which is known to not existing, such as `non-existent.test.dnssec-tools.org`. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC record.

Note: The query issued in this test MUST be sent to a NSEC signed zone. Getting back appropriate NSEC3 records does not indicate a failure, but a bad test.

3.1.10. Supports negative answers with NSEC3 records

Purpose: This tests whether a resolver properly returns NSEC3 records ([RFC5155]) for a non-existing domain in a DNSSEC signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an A record which is known to be non-existent, such as non-existent.nsec3-ns.test.dnssec-tools.org. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC3 record.

Note: The query issued in this test MUST be sent to a NSEC3 signed zone. Getting back appropriate NSEC records does not indicate a failure, but a bad test.

3.1.11. Supports queries where DNAME records lead to an answer

Purpose: This tests whether a resolver can query for an A record in a zone with a known DNAME referral for the record's parent.

Test: Send a request to the resolver under test for an A record which is known to exist in a signed zone within a DNAME referral child zone, such as good-a.dname-good-ns.test.dnssec-tools.net.

SUCCESS: A DNS response was received that contains a DNAME in the answer section. An RRSIG MUST also be received in the answer section that covers the DNAME record.

3.1.12. Permissive DNSSEC

Purpose: To see if a validating resolver is ignoring DNSSEC validation failures.

Pre-requisite: Supports the AD bit.

Test: ask for data from a broken DNSSEC delegation such as badsign-a.test.dnssec-tools.org.

SUCCESS: A reply with the Rcode set to SERVFAIL

3.1.13. UDP size limits

TBD

3.1.14. Supports Unknown RRtypes

Purpose: Some DNS Resolvers/gateways only support some RRtypes. This causes problems for applications that need recently defined types.

Pre-requisite: "Supports UDP or TCP".

Test: Send a request for recently defined type or unknown type in the 20000-22000 range, that resolves to a server that will return answer for all types, such as alltypes.res.dnssecready.org

SUCCESS: A DNS response was retrieved that contains the type requested in the answer section.

3.2. Direct Network Queries

If need be, a Host Validator may need to make direct queries to authoritative servers or known Open Recursive Resolvers in order to collect data. To do that, a number of key network features MUST be functional.

3.2.1. Support for Remote UDP Over Port 53

Purpose: This tests basic UDP functionality to outside the local network.

Test: A DNS request is sent to a known distant authoritative server for a record known to be within that server's authoritative data.
Example: send a query to the address of ns1.dnssec-tools.org for the www.dnssec-tools.org/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: an implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but does not need to be.

3.2.2. Support for Remote UDP With Fragmentation

Purpose: This tests if the local network can receive fragmented UDP answers

Pre-requisite: Local UDP > 1500 is possible

Test: A DNS request is sent over UDP to a known distant DNS address asking for a record that has answer larger than 2000 bytes. Example send a query for the dnssec-tools.org/DNSKEY record with the DO bit set in the outgoing query.

Success: A DNS response was received that contains the large answer.

Note: A failure in getting large answers over UDP is not a serious problem if TCP is working.

3.2.3. Support for Outbound TCP Over Port 53

Purpose: This tests basic TCP functionality to outside the local network.

Test: A DNS request is sent over TCP to a known distant authoritative server for a record known to be within that server's authoritative data. Example: send a query to the address of ns1.dnssec-tools.org for the www.dnssec-tools.org/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: an implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but does not need to be.

4. Aggregating The Results

Some conclusions can be drawn from the results of the above tests in an "aggregated" form. This section defines some labels to assign to a resolver under test given the results of the tests run against them.

4.1. Resolver capability description

This section will group and label certain common results TBD

Resolvers are classified into following broad behaviors:

Validator: The resolver passes all DNSSEC tests and had the AD bit appropriately set.

DNSSEC Aware: The resolver passes all DNSSEC tests, but does not appropriately set the AD bit on answers, indicating it is not validating. A Host Validator will function fine using this resolver as a forwarder.

Non-DNSSEC capable: The resolver is not DNSSEC aware and will make it hard for a Host Validator to operate behind it. It MAY be usable for querying for data that is in known insecure sections of the DNS tree.

Not a DNS Resolver: This is a bad address and not used anymore.

While it would be great if all resolvers fell cleanly into one of the broad categories above, that is not the case. For that reason it is

necessary to augment the classification with more descriptive result, this is done by adding the word "Partial" in front of Validator/DNSSEC Aware classifications, followed by sub-descriptors of what is not working.

Unknown: Failed Unknown test

DNAME: Failed DNAME test

NSEC3: Failed NSEC3 test

TCP: TCP not available

SlowBig: UDP is size limited but TCP fallback works

NoBig: TCP not available and UDP is size limited

Permissive: Passes data known to fail validation

5. Roadblock Avoidance

[Editors note: the goal of this document is to tie the above tests and aggregations to avoidance practices; however right now the "tie" part of this text is known to be weak. The goal of this document is specifically to improve this tie as work on this document progresses]

Once we have determined what level of support is available in the neighboring network, we can determine what MUST be done in order to effectively act as a validating resolver. This section discusses some of the options available given the results from the previous sections.

The general fallback approach can be described by the following sequence:

If the resolver is labeled as "Validator" or "DNSSEC aware"

Send query through this resolver and perform local validation on the results.

If validation fails, try the next resolver

Else if the resolver is labeled "Not a DNS Resolver" or "Non-DNSSEC capable"

Mark it as unusable and try next resolver

Else if no more resolvers are configured and if direct queries are supported try iterating from Root

Else return a useful error code

While attempting resolution through a particular recursive name server with a particular transport method that worked, any transport-specific parameters MUST be remembered in order to short-circuit any unnecessary fallback attempts.

Transport-specific parameters MUST also be remembered for each authoritative name server that is queried while performing an iterative mode lookup.

Any transport settings that are remembered for a particular name server MUST be periodically refreshed; they should also be refreshed when an error is encountered as described below.

For a stub resolver, problems with the name server MAY manifest themselves as the following types of error conditions:

- o No response/error response or missing DNSSEC meta-data.
- o Illegal Response, which prevents the validator from fetching all necessary records required for constructing an authentication chain. This could result when referral loops are encountered, when any of the antecedent zone delegations are lame, when aliases are erroneously followed for certain RRtypes (such as SOA, DNSKEYs or DS records), or when resource records for certain types (e.g. DS) are returned from a zone that is not authoritative for such records.
- o Bogus Response, when the cryptographic assertions in the authentication chain do not validate properly.

For each of the above error conditions a validator MAY adopt the following dynamic fallback technique, preferring a particular approach if it is known to work for a given name server or zone from previous attempts.

- o No response, error response, or missing DNSSEC meta-data
 - * Re-try with different EDNS0 sizes (4096, 1492, None)
 - * Re-try with TCP only

- * Perform an iterative query starting from Root if the previous error was returned from a lookup that had recursion enabled.
- * Re-try using an alternative transport method, if this alternative method is known (configured) to be supported by the nameserver in question.
- o Illegal Response
 - * Perform an iterative query starting from Root if the previous error was returned from a lookup that had recursion enabled.
 - * Check if any of the antecedent zones up to the closest configured trust anchor are provably insecure.
- o Bogus Response
 - * Perform an iterative query starting from Root if the previous error was returned from a lookup that had recursion enabled.

For each fallback technique, attempts to multiple potential name servers should be skewed such that the next name server is tried when the previous one encounters an error or a timeout is reached, whichever is earlier.

The validator SHOULD remember, in its zone-specific fallback cache, any broken behavior identified for a particular zone for a duration of that zone's SOA negative TTL.

The validator MAY place name servers that exhibit broken behavior into a blacklist, and bypass these name servers for all zones that they are authoritative for. The validator MUST time out entries in this name server blacklist periodically, where this interval could be set to be the same as the DNSSEC BAD cache default TTL.

5.1. Partial Resolver Usage

It MAY be possible to use Non-DNSSEC Capable caching resolvers in careful ways if maximum optimization is desired. This section describes some of the advanced techniques that could be used to use a resolver in at least a minimal way. Most of the time this would be unnecessary, except in the case where none of the resolvers are fully compliant and thus the choices would be to use them at least minimally or not at all (and no caching benefits would be available).

5.1.1. Known Insecure Lookups

If a resolver is Non-DNSSEC Capable but a section of the DNS tree has been determined to be Provably Insecure [RFC4035], then queries to this section of the tree MAY be sent through Non-DNSSEC Capable caching resolver.

5.1.2. Partial NSEC/NSEC3 Support

TBD

6. Start-Up and Network Connectivity Issues

A number of scenarios will produce either short-term or long-term connectivity issues with respect to DNSSEC validation. Consider the following cases:

Time Synchronization: Time synchronization problems can occur when a device which has been off for a period of time and the clock is no longer in close synchronization with "real time" or when a device always has clock set to the same time during start-up. This will cause problems when the device needs to resolve their source of time synchronization, such as "ntp.example.com".

Changing Network Properties: A newly established network connection MAY change state shortly after a HTTP-based pay-wall authentication system has been used. This is especially common in hotel networks, where DNSSEC, validation and even DNS are not functional until the user proceeds through a series of forced web pages used to enable their network. The tests in Section 3 will produce very different results before and after the network authorization has succeeded. APIs exist on many operating systems to detect initial network device status changes, such as right after DHCP has finished, but few (none?) exist to detect that authentication through a pay-wall has succeeded.

There are only two choices when situations like this happen:

Continue to perform DNSSEC processing, which will likely result in all DNS requests failing. This is the most secure route, but causes the most operational grief for users.

Turn off DNSSEC support until the network proves to be usable. This allows the user to continue using the network, at the sacrifice of security. It also allows for a denial of security-service attack if a man-in-the-middle can convince a device that DNSSEC is impossible.

6.1. What To Do

TBD

7. IANA Considerations

No IANA actions are require to support this document

8. Security Considerations

This document discusses problems that may occur while deploying the secure DNSSEC protocol and what mitigation's can be used to help detect and mitigate these problems. Following these suggestions will result in a more secure DNSSEC operational environment than if DNSSEC was simply disabled when it fails to perform as expected.

9. Acknowledgments

Multiple lessons learned from multiple implementations led to the development of this document, including (in alphabetical order) DNSSEC-Tools' DNSSEC-Check, DNSSEC_Resolver_Check, dnssec-trigger, FCC_Grade.

The following people contributed to portions of this document in some fashion:

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

Authors' Addresses

Wes Hardaker
Parsons
P.O. Box 382
Davis, CA 95617
US

Email: hardaker@tislabs.com

Olafur Gudmundsson
Shinkuro Inc.
4922 Fairmont Av, Suite 250
Bethesda, MD 20814
USA

Email: ogud@ogud.com

Suresh Krishnaswamy
Parsons
7110 Samuel Morse Dr
Columbia, MD 21046
US

Email: suresh@tislabs.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 28, 2015

J. Abley
Dyn, Inc.
B. Dickson
Twitter, Inc.
W. Kumari
Google
G. Michaelson
APNIC
November 24, 2014

AS112 Redirection using DNAME
draft-ietf-dnsop-as112-dname-06

Abstract

AS112 provides a mechanism for handling reverse lookups on IP addresses that are not unique (e.g., RFC 1918 addresses). This document describes modifications to the deployment and use of AS112 infrastructure that will allow zones to be added and dropped much more easily, using DNAME resource records.

This approach makes it possible for any DNS zone administrator to sink traffic relating to parts of the global DNS namespace under their control to the AS112 infrastructure without coordination with the operators of AS112 infrastructure.

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 May 28, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Design Overview	4
3. AS112 Operations	4
3.1. Extensions to Support DNAME Redirection	4
3.2. Redirection of Query Traffic to AS112 Servers	5
4. Continuity of AS112 Operations	5
5. Candidate Zones for AS112 Redirection	6
6. DNAME Deployment Considerations	6
7. IAB Statement Regarding this .ARPA Request	7
8. IANA Considerations	7
8.1. Address Assignment	7
8.2. Hosting of AS112.ARPA	9
8.3. Delegation of AS112.ARPA	10
9. Security Considerations	10
10. Acknowledgements	10
11. References	11
11.1. Normative References	11
11.2. Informative References	11
Appendix A. Assessing Support for DNAME in the Real World . . .	12
A.1. Methodology	12
A.2. Results	14
Appendix B. Editorial Notes	14
B.1. Change History	14
Authors' Addresses	15

1. Introduction

Many sites connected to the Internet make use of IPv4 addresses that are not globally unique. Examples are the addresses designated in [RFC1918] for private use within individual sites.

Devices in such environments may occasionally originate Domain Name System (DNS) queries (so-called "reverse lookups") corresponding to those private-use addresses. Since the addresses concerned have only local significance, it is good practice for site administrators to ensure that such queries are answered locally. However, it is not uncommon for such queries to follow the normal delegation path in the public DNS instead of being answered within the site.

It is not possible for public DNS servers to give useful answers to such queries. In addition, due to the wide deployment of private-use addresses and the continuing growth of the Internet, the volume of such queries is large and growing. The AS112 project aims to provide a distributed sink for such queries in order to reduce the load on the IN-ADDR.ARPA authoritative servers. The AS112 project is named after the Autonomous System Number (ASN) that was assigned to it.

Prior to implementation of this technique, the AS112 project did not accommodate the addition and removal of DNS zones elegantly. Since additional zones of definitively local significance are known to exist, this presents a problem. This document describes modifications to the deployment and use of AS112 infrastructure that will allow zones to be added and dropped much more easily.

The AS112 project is described in detail in [I-D.ietf-dnsop-rfc6304bis].

The AS112 nameservers (PRISONER.IANA.ORG, BLACKHOLE-1.IANA.ORG and BLACKHOLE-2.IANA.ORG) are required to answer authoritatively for each and every zone that is delegated to them. If a zone is delegated to AS112 nameservers without those nameservers being configured ahead of time to answer authoritatively for that zone, there is a detrimental impact on clients following referrals for queries within that zone. This misconfiguration is colloquially known as a "lame delegation".

AS112 nameserver operators are only loosely-coordinated, and hence adding support for a new zone (or, correspondingly, removing support for a zone that is no longer delegated to the AS112 nameservers) is difficult to accomplish with accuracy. Testing AS112 nameservers remotely to see whether they are configured to answer authoritatively for a particular zone is similarly challenging since AS112 nodes are distributed using anycast [RFC4786].

This document defines a more flexible approach for sinking queries on AS112 infrastructure that can be deployed alongside unmodified, existing AS112 nodes. Instead of delegating additional zones directly to AS112 nameservers, DNAME [RFC6672] redirection is used. This approach has the advantage that query traffic for arbitrary parts of the namespace can be directed to AS112 servers without those

servers having to be reconfigured every time a zone is added or removed.

This approach makes it possible for any DNS zone administrator to sink traffic relating to parts of the global DNS namespace under their control to the AS112 infrastructure without coordination with the operators of AS112 infrastructure.

2. Design Overview

A new zone, EMPTY.AS112.ARPA, is delegated to a single nameserver BLACKHOLE.AS112.ARPA (IPv4 address TBAv4-1, IPv6 address TBAv6-1).

The IPv4 address TBAv4-1 has been assigned by the IANA such that the address is coverable by a single IPv4 /24 prefix, and that no other address covered by that prefix is in use. The IPv6 address TBAv6-1 has been similarly assigned such that no other address within a covering /48 is in use. This addressing plan accommodates the anycast distribution of the BLACKHOLE.AS112.ARPA service using a single IPv4 service prefix and a single IPv6 service prefix. See [RFC4786] for more discussion of anycast service distribution; see Section 8 for the specific requests this document makes of the IANA.

Some or all of the existing AS112 nodes SHOULD be extended to support these new nameserver addresses, and to host the EMPTY.AS112.ARPA zone. See [I-D.ietf-dnsop-rfc6304bis] for revised guidance to AS112 server operators.

Each part of the DNS namespace for which it is desirable to sink queries at AS112 nameservers should be redirected to the EMPTY.AS112.ARPA zone using DNAME [RFC6672]. See Section 3.2 for guidance to zone administrators.

3. AS112 Operations

3.1. Extensions to Support DNAME Redirection

Guidance to operators of AS112 nodes is extended to include configuration of the TBAv4-1, and TBAv6-1 addresses, and the corresponding announcement of covering routes for those addresses, and to host the EMPTY.AS112.ARPA zone.

IPv4-only AS112 nodes should only configure the TBAv4-1 nameserver address; IPv6-only AS112 nodes should only configure the TBAv6-1 nameserver address.

It is only necessary for a single AS112 server operator to implement these extensions for this mechanism to function as intended. It is

beneficial if many more than one AS112 server operators make these changes, however, since that provides for greater distribution and capacity for the nameservers serving the EMPTY.AS112.ARPA zone. It is not necessary for all AS112 server operators to make these changes for the mechanism to be viable.

Detailed instructions for the implementation of these extensions is included in [I-D.ietf-dnsop-rfc6304bis].

3.2. Redirection of Query Traffic to AS112 Servers

Once the EMPTY.AS112.ARPA zone has been deployed using the nameservers described in Section 3.1, redirections may be installed in the DNS namespace for queries that are intended to be answered by the AS112 infrastructure.

For example, reverse queries corresponding to TEST-NET-1 (192.0.2.0/24) [RFC5737] could be redirected to AS112 nameservers by installing a DNAME resource record in the 192.IN-ADDR.ARPA zone, as illustrated in Figure 1.

```
$ORIGIN 192.IN-ADDR.ARPA.  
...  
2.0      IN      DNAME    EMPTY.AS112.ARPA.  
...
```

Figure 1

There is no practical limit to the number of redirections that can be configured in this fashion. Redirection of a particular part of the namespace to EMPTY.AS112.ARPA can be removed at any time, under the control of the administrators of the corresponding part of the DNS namespace. No changes to deployed AS112 nodes incorporating the extensions described in this document are required to support additional redirections. A list of possible candidates for AS112 redirection can be found in Section 5.

DNAME resource records deployed for this purpose can be signed with DNSSEC [RFC4033], providing a secure means of authenticating the legitimacy of each redirection.

4. Continuity of AS112 Operations

Existing guidance to AS112 server operators to accept and respond to queries directed at the PRISONER.IANA.ORG, BLACKHOLE-1.IANA.ORG and BLACKHOLE-2.IANA.ORG nameservers should continue to be followed, and no changes to the delegation of existing zones hosted on AS112 servers should occur. These measures are intended to provide

continuity of operations for zones currently delegated to AS112 servers and avoid any accidental client impact due to the changes proposed in this document.

Once it has become empirically and quantitatively clear that the EMPTY.AS112.ARPA zone is well-hosted to the extent that it is thought that the existing, unmodified AS112 servers host 10.IN-ADDR.ARPA, the decision might be made to replace the delegation of those [RFC1918] zones with DNAME redirection. Once implemented, the PRISONER.IANA.ORG, BLACKHOLE-1.IANA.ORG and BLACKHOLE-2.IANA.ORG nameservers could be retired. This document gives no such direction to the IANA, however.

5. Candidate Zones for AS112 Redirection

All zones listed in [RFC6303] are candidates for AS112 redirection.

Since no pre-provisioning is required on the part of AS112 operators to facilitate sinking of any name in the DNS namespace by AS112 infrastructure, this mechanism supports AS112 redirection by any zone owner in the DNS.

This document is simply concerned with provision of the AS112 redirection service, and does not specify that any particular AS112 redirection be put in place.

6. DNAME Deployment Considerations

DNAME was specified years after the original implementations of [RFC1035], and hence universal deployment cannot be expected. [RFC6672] specifies a fall-back mechanism which makes use of synthesised CNAME RRsets for this reason. The expectation that design choices in the DNAME specification ought to mitigate any lack of deployment is reviewed below. Experimental validation of those expectations is included in Appendix A.

It is a fundamental design requirement of AS112 service that responses be cached. We can safely declare DNAME support on the authoritative server to be a prerequisite for DNAME redirection, but the cases where individual elements in resolver chains do not support DNAME processing deserve closer examination.

The expected behaviour when a DNAME response is supplied to a resolver that does not support DNAME is that the accompanying, synthesised CNAME will be accepted and cached. Re-query frequency will be determined by the TTLs returned by the DNAME-responding authoritative servers.

Resolution of the CNAME target is straightforward and functions exactly as the AS112 project has operated since it was deployed. The negative caching [RFC2308] of the CNAME target follows the parameters defined in the target zone, EMPTY.AS112.ARPA. This has the side-effects that all redirected names ultimately landing on an AS112 node will be negatively-cached with the same parameters, but this lack of flexibility seems non-controversial; the effect of reducing the negative cache TTL would be increased query volume on the AS112 node operator concerned, and hence controls seem well-aligned with operation.

Validating resolvers (i.e. those requesting and processing DNSSEC [RFC4033] metadata) are required to implement DNAME, and hence should not make use of synthesised CNAME RRs. The lack of signature over a received CNAME RR should hence not limit the ability to sign the redirection point, and for those signatures to be validated.

In the case where a recursive server implements DNAME, but DNAME is not implemented in a stub resolver, CNAME synthesis will again provide a viable path.

DNAME support on AS112 nodes themselves is never required under this proposal.

7. IAB Statement Regarding this .ARPA Request

With the publication of this document, the IAB approves of the delegation of 'AS112' in the ARPA domain. Under [RFC3172], the IAB has requested that IANA delegate and provision "AS112.ARPA" as specified in this specification. However, the IAB does not take any architectural or technical position about this specification.

8. IANA Considerations

8.1. Address Assignment

This document requests that IANA assign IPv4 and IPv6 number resources in conformance with section 4 of [RFC2860].

The IANA is requested to assign one IPv4 /24 netblock and register its use in the IPv4 Special-Purpose Address Registry [RFC6890] as follows:

Name	Value
Address Block	As determined by IANA
Name	AS112-v4
RFC	[THIS DOCUMENT]
Allocation Date	As determined by IANA
Termination Date	N/A
Source	True
Destination	True
Forwardable	True
Global	True
Reserved-by-Protocol	False

We suggest that IANA assign 192.31.196.0/24 from the IPv4 Recovered Address Space Registry, but any /24 which has been unassigned and unadvertised for at least twelve months is acceptable.

The IANA is requested to assign one IPv6 /48 netblock and register its use in the IPv6 Special-Purpose Address Registry [RFC6890] as follows:

Name	Value
Address Block	As determined by IANA
Name	AS112-v6
RFC	[THIS DOCUMENT]
Allocation Date	As determined by IANA
Termination Date	N/A
Source	True
Destination	True
Forwardable	True
Global	True
Reserved-by-Protocol	False

We suggest that IANA assign 2001:112::/48 from the IETF Protocol Assignments allocation [RFC2928], but /48 which has been unassigned and unadvertised for at least twelve months is acceptable.

Once assigned, all occurrences of TBAv4 in this document should be replaced by the IPv4 netblock assigned, in conventional notation. Occurrences of TBAv4-1 should be replaced with an address from the netblock with lowest octet set to 1. Similarly, all occurrences of TBAv6 in this document should be replaced by the IPv6 netblock assigned, in conventional notation, and TBAv6-1 replaced with an address from that netblock with the lowest 48 bits set to the value 1. Once those changes are made, this paragraph may be removed prior to publication.

The netblocks assigned by the IANA for this purpose are TBAv4 and TBAv6.

8.2. Hosting of AS112.ARPA

The IANA is requested to host and sign the zone AS112.ARPA using nameservers and DNSSEC signing infrastructure of their choosing, as shown in Figure 2. SOA RDATA may be adjusted by the IANA to suit their operational requirements.

```

$ORIGIN AS112.ARPA.
$TTL 3600

@      IN      SOA      BLACKHOLE.AS112.ARPA. NOC.DNS.ICANN.ORG. (
                                1              ; serial
                                10800           ; refresh
                                3600            ; retry
                                1209600        ; expire
                                3600 )          ; negative cache TTL

                                NS      A.IANA-SERVERS.NET.
                                NS      B.IANA-SERVERS.NET.
                                NS      C.IANA-SERVERS.NET.

BLACKHOLE      A      TBAv4-1
                AAAA   TBAv6-1

HOSTNAME        NS      BLACKHOLE

EMPTY           NS      BLACKHOLE

```

Figure 2

8.3. Delegation of AS112.ARPA

Once the AS112.ARPA zone is being hosted in production, the IANA is requested to arrange delegation from the ARPA zone according to normal IANA procedure for ARPA zone management, to the nameservers used in carrying out the direction in Section 8.2. The whois contact information for the new record should be specified by the IAB under [RFC3172].

9. Security Considerations

This document presents no known additional security concerns to the Internet.

For security considerations relating to AS112 service in general, see [I-D.ietf-dnsop-rfc6304bis].

10. Acknowledgements

The authors acknowledge the valuable contributions of Bob Harold and other participants in the DNSOP working group in the preparation of this document.

11. References

11.1. Normative References

- [I-D.ietf-dnsop-rfc6304bis]
Abley, J. and W. Maton, "AS112 Nameserver Operations",
draft-ietf-dnsop-rfc6304bis-04 (work in progress), July
2014.
- [RFC1035] Mockapetris, P., "Domain names - implementation and
specification", STD 13, RFC 1035, November 1987.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS
NCACHE)", RFC 2308, March 1998.
- [RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the
DNS", RFC 6672, June 2012.

11.2. Informative References

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
E. Lear, "Address Allocation for Private Internets", BCP
5, RFC 1918, February 1996.
- [RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of
Understanding Concerning the Technical Work of the
Internet Assigned Numbers Authority", RFC 2860, June 2000.
- [RFC2928] Hinden, R., Deering, S., Fink, R., and T. Hain, "Initial
IPv6 Sub-TLA ID Assignments", RFC 2928, September 2000.
- [RFC3172] Huston, G., "Management Guidelines & Operational
Requirements for the Address and Routing Parameter Area
Domain ("arpa")", BCP 52, RFC 3172, September 2001.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S.
Rose, "DNS Security Introduction and Requirements", RFC
4033, March 2005.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast
Services", BCP 126, RFC 4786, December 2006.
- [RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks
Reserved for Documentation", RFC 5737, January 2010.
- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC
6303, July 2011.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman,
"Special-Purpose IP Address Registries", BCP 153, RFC
6890, April 2013.

Appendix A. Assessing Support for DNAME in the Real World

To measure the extent to which the DNAME construct is supported in the Internet, we have used an experimental technique to test the DNS resolvers used by end hosts, and derive from the test a measurement of DNAME support within the Internet.

A.1. Methodology

The test was conducted by loading a user's browser with 4 URLs to retrieve. The first three comprise the test setup, while the final URL communicates the result to the experiment controller. The URLs are:

- A `http://a.<unique_string>.dname.example.com/1x1.png?`
`a.<unique_string>.dname`
- B `http://b.dname.example.com/1x1.png?`
`b.<unique_string>.dname`
- C `http://c.<unique_string>.target.example.net/1x1.png?`
`c.<unique_string>.target`
- D `http://results.recorder.example.net/1x1.png?`
`results.<unique_string>?za=<a_result>&zb=<b_result>&zc=<c_result>`

The A URL is designed to test the end users capability to resolve a name that has never been seen before, so that the resolution of this domain name will reliably result in a query at the authoritative name server. This is intended to test the use of domain names where there is a dynamic component that also uses the DNAME construct.

The B URL is deliberately designed to be cached by caching resolvers that are used in the process of resolving the domain name.

The C URL is a control URL. This is a unique URL, similar to A, but does not refer to a DNAME structure.

The D URL uses a static cacheable domain name.

The `<unique_string>` value is common to the four URLs used in each individual instance of this test, but varies from test to test. The result is that each end user is presented with a unique string.

The contents of the EXAMPLE.COM, TARGET.EXAMPLE.NET and RECORDER.EXAMPLE.NET zones are shown in Figure 3.

```
$ORIGIN EXAMPLE.COM.
...
DNAME.                IN  DNAME  TARGET.EXAMPLE.NET.
...

$ORIGIN TARGET.EXAMPLE.NET.
...
B                      IN  A      192.0.2.0
*                      IN  A      192.0.2.0
...

$ORIGIN RECORDER.EXAMPLE.NET.
...
RESULTS                IN  A      192.0.2.0
...
```

Figure 3

The first three URLs (A, B and C) are loaded as tasks into the user's browser upon execution of the test's script. The script starts a timer with each of these URLs to measure the elapsed time to fetch the URL. The script then waits for the three fetches to complete, or 10 seconds, whichever occurs first. The script then loads the results of the three timers into the GET arguments of the D URL, and performs a fetch to pass these results back to the experiment's server.

Logs on the web server reached at RESULTS.EXAMPLE.NET will include entries of the form shown in Figure 4. If any of the URLs fail to load within 10 seconds the D URL will report the failure as a "null" timer value.

```
GET /1x1.png?results.<unique_string>?za=1822&zb=1674&zc=1582
GET /1x1.png?results.<unique_string>?za=null&zb=null&zc=161
```

Figure 4

The script has been encoded in Adobe Flash with a simple image in the form of an online advertisement. An online advertisement network has been used to distribute the script. The script is invoked when the advertisement is presented in the end user's browser or application, and does not require the user to click on the supplied image in any way. The advertisement placement parameters were set to to broadest possible scope to sample users from across the entire internet.

A.2. Results

The test was loaded into an advertisement distributed on 2013-10-10 and 2013-10-11.

	Count	Percentage
Recorded Results:	338,478	
A or B Loaded:	331,896	98.1%
A Fail and B Fail:	6,492	1.9%
A Fail and B Load:	4,249	1.3%
A Load and B Fail:	1,624	0.5%
C Fail:	9,355	2.8%

Table 1

These results indicate that at most 1.9% of tested clients use DNS resolvers that fail to resolve a domain name that contains a DNAME redirection. However the failure rate of slightly lower than 3% for the control URL indicates that the failure rate for the DNAME construct lies within the bounds of error within the experimental framework. We conclude that there is no evidence of a consistent failure on the part of deployed DNS resolvers to correctly resolve a DNAME construct.

This experiment was conducted by Geoff Huston and George Michaelson.

Appendix B. Editorial Notes

This section (and sub-sections) to be removed prior to publication.

B.1. Change History

00 Initial write-up of Brian's idea, circulated for the purposes of entertainment.

01 Some particularly egregious spelling mistakes fixed. Warren Kumari and George Michaelson added as co-authors. Intended status changed to informational. Appendix on DNAME testing added, describing an experiment conducted by Geoff Huston and George Michaelson.

- 00 Adopted by dnsop in IETF88, Vancouver; resubmitted as draft-ietf-dnsop-as112-dname. Changed contact info for Brian.
- 01 Minor updates following submission of draft-jabley-dnsop-rfc6304bis.
- 02 Text in IANA Considerations section dealing with address assignments modified following informal advice received from Leo Vegoda.
- 03 Updated references to 6304 following guidance from working group chairs.
- 04 Corrected an error picked up by Bob Harold.
- 05 Addressed various comments from the IESG and IAB. Updated Brian's contact info. Minor spelling and grammatical corrections. Added text to the abstract and introduction to reinforce the point that this approach allows liberal use of AS112 infrastructure without coordination with AS112 operators.
- 06 Made changes requested by the IAB relating to [RFC3172].

Authors' Addresses

Joe Abley
Dyn, Inc.
186 Albert Street, Suite 103
London, ON N6A 1M1
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

Brian Dickson
Twitter, Inc.

Email: bdickson@twitter.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Email: warren@kumari.net

George Michaelson
APNIC

Email: ggm@apnic.net

DNSOP
Internet-Draft
Intended status: Standards Track
Expires: July 10, 2015

W. Hardaker
Parsons, Inc.
January 6, 2015

Child To Parent Synchronization in DNS
draft-ietf-dnsop-child-synchronization-07

Abstract

This document specifies how a child zone in the DNS can publish a record to indicate to a parental agent that the parental agent may copy and process certain records from the child zone. The existence of the record and any change in its value can be monitored by a parental agent and acted on depending on local policy.

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 July 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology Used in This Document	3
2. Definition of the CSYNC RRTYPE	4
2.1. The CSYNC Resource Record Format	4
2.1.1. The CSYNC Resource Record Wire Format	5
2.1.2. The CSYNC Presentation Format	6
2.1.3. CSYNC RR Example	6
3. CSYNC Data Processing	7
3.1. Processing Procedure	7
3.2. CSYNC Record Types	8
3.2.1. The NS type	9
3.2.2. The A and AAAA types	9
4. Operational Considerations	10
4.1. Error Reporting	10
4.2. Child Nameserver Selection	10
4.3. Out-of-balliwick NS Records	11
4.4. Documented Parental Agent Type Support	11
4.5. Removal of the CSYNC records	12
4.6. Parent/Child/Grandchild Glue Synchronization	12
5. Security Considerations	12
6. IANA Considerations	13
7. Acknowledgments	14
8. References	14
8.1. Normative References	14
8.2. Informative References	14
Author's Address	15

1. Introduction

This document specifies how a child zone in the DNS ([RFC1034], [RFC1035]) can publish a record to indicate to a parental agent (see section Section 2 for a definition of "parental agent") that it can copy and process certain records from the child zone. The existence of the record and any change in its value can be monitored by a parental agent and acted on depending on local policy.

Currently today, some resource records (RRs) in a parent zone are typically expected to be in sync with the source data in the child's zone. The most common records that should match are the nameserver (NS) records and any necessary associated address records (A and AAAA), also known as "glue records". These records are referred to as "delegation records".

It has been challenging for operators of child DNS zones to update their delegation records within the parent's set in a timely fashion. These difficulties may stem from operator laziness, as well as from the complexities of maintaining a large number of DNS zones. Having an automated mechanism for signaling updates will greatly ease the child zone operator's maintenance burden and improve the robustness of the DNS as a whole.

This draft introduces a new Resource Record Type (RRType) named "CSYNC" that indicates which delegation records published by a child DNS operator should be processed by a parental agent and used to update the parent zone's DNS data.

This specification was not designed to synchronize DNSSEC security records, such as DS RRsets. For a solution to this problem, see the complementary solution [RFC7344], which is designed to maintain security delegation information. In addition, this specification does not address how to perform bootstrapping operations, including to get the required initial DNSSEC-secured operating environment in place.

1.1. Terminology Used in This Document

The terminology used in this document is defined in this section.

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 [RFC2119].

Terminology describing relationships between the interacting roles involved in this document are defined in the following list:

Child: The entity on record that has the delegation of the domain from the parent

Parent: The domain in which the child is registered

Child DNS operator: The entity that maintains and publishes the zone information for the child DNS

Parental agent: The entity that the child has relationship with, to change its delegation information

2. Definition of the CSYNC RRTYPE

The CSYNC RRTYPE contains, in its RDATA component, these parts: an SOA serial number, a set of flags and a simple bit-list indicating the DNS RRTYPES in the child that should be processed by the parental agent in order to modify the DNS delegation records within the parent's zone for the child DNS operator. Child DNS operators wanting a parental agent to perform the synchronization steps outlined in this document MUST publish a CSYNC record at the apex of the child zone. Parental agent implementations MAY choose to query child zones for this record and process DNS record data as indicated by the Type Bit Map field in the RDATA of the CSYNC record. How the data is processed is described in Section 3.

Parental agents MUST process the entire set of child data indicated by the Type Bit Map field (i.e., all record types indicated along with all of the necessary records to support processing of that type) or else parental agents MUST NOT make any changes to parental records at all. Errors due to unsupported Type Bit Map bits, or otherwise nonpunishable data, SHALL result in no change to the parent zone's delegation information for the Child. Parental agents MUST ignore a Child's CSYNC RDATA set if multiple CSYNC resource records are found; only a single CSYNC record should ever be present.

The parental agent MUST perform DNSSEC validation ([RFC4033], [RFC4034], [RFC4035]), of the CSYNC RRTYPE data and MUST perform DNSSEC validation of any data to be copied from the Child to the Parent. Parents MUST NOT process any data from any of these records if any of the validation results indicate anything other than "Secure" [RFC4034] or if any the required data can not be successfully retrieved.

2.1. The CSYNC Resource Record Format

flags defined in this document are as follows:

0x00 0x01: "immediate"

0x00 0x02: "soaminimum"

The definitions for how the flags are to be used can be found later in Section Section 3.

The remaining flags are reserved for use by future specifications. Undefined flags MUST be set to 0 by CSYNC publishers. Parental agents MUST NOT process a CSYNC record if it contains a 1 value for a flag that is unknown to or unsupported by the parental agent.

2.1.1.2.1. The Type Bit Map Field

The Type Bit Map field indicates the record types to be processed by the parental agent, according to the procedures in Section Section 3. The Type Bit Map field is encoded in the same way as the Type Bit Maps field of the NSEC record, described in [RFC4034], Section 4.1.2. If a bit has been set that a parental agent implementation does not understand, the parental agent MUST NOT act upon the record. Specifically: a parental agent must not just copy the data and must understand the semantics associated with an bit in the Type Bit Map field that has been set to 1.

2.1.1.2. The CSYNC Presentation Format

The CSYNC presentation format is as follows:

The SOA Serial field is represented as an integer.

The Flags field is represented as an integer.

The Type Bit Map field is represented as a sequence of RR type mnemonics. When the mnemonic is not known, the TYPE representation described in [RFC3597], Section 5, MUST be used. Implementations that support parsing of presentation format records SHOULD be able to read and understand these TYPE representations as well.

2.1.1.3. CSYNC RR Example

The following CSYNC RR shows an example entry for "example.com" that indicates the NS, A and AAAA bits are set and should be processed by the parental agent for example.com. The parental agent should pull data only from a zone using a minimum SOA serial number of 66 (0x42 in hexadecimal).

example.com. 3600 IN CSYNC 66 3 A NS AAAA

The RDATA component of the example CSYNC RR would be encoded on the wire as follows:

```
0x00 0x00 0x00 0x42      (SOA Serial)
0x00 0x03                (Flags = immediate | soaminimum)
0x00 0x04 0x60 0x00 0x00 0x08 (Type Bit Map)
```

3. CSYNC Data Processing

The CSYNC record and associated data must be processed as an "all or nothing" operation set. If a parental agent fails to successfully query for any of the required records, the whole operation MUST be aborted. (Note that a query resulting in "no records exist" as proven by NSEC or NSEC3 is to be considered successful).

Parental agents MAY:

Process the CSYNC record immediately if the "immediate" flag is set. If the "immediate" flag is not set, the parental agent MUST NOT act until the zone administrator approves the operation through an out-of-band mechanism (such as through pushing a button via a web interface).

Choose not to process the CSYNC record immediately, even if the "immediate" flag is set. That is, a parental agent might require the child zone administrator approve the operation through an out-of-band mechanism (such as through pushing a button via a web interface).

Note: how the approval is done out-of-band is outside the scope of this document and is implementation-specific to parental agents.

3.1. Processing Procedure

The following shows a sequence of steps that SHOULD be used when collecting and processing CSYNC records from a child zone. Because DNS queries are not allowed to contain more than one "question" at a time, a sequence of requests is needed. When processing a CSYNC transaction request, all DNS queries should be sent to a single authoritative name server for the child zone. To ensure a single host is being addressed, DNS over TCP SHOULD be used to avoid conversing with multiple nodes at an anycast address.

1. Query for the child zone's SOA record

2. Query for the child zone's CSYNC record
3. Query for the child zone's data records, as required by the CSYNC record's Type Bit Map field
 - * Note: if any of the resulting records being queried are not authoritative within the child zone but rather in a grandchild or deeper, SOA record queries must be made for the grandchildren. This will require the parental agent to determine where the child/grandchild zone cuts occur. Because of the additional operational complexity, parental agents MAY choose not to support this protocol with children making use of records that are authoritative in the grandchildren.
4. Query for the collected SOA records again, starting with the deepest and ending with the SOA of the child's.

If the SOA records from the first, middle and last steps for a given zone have different serial numbers (for example, because the zone was edited and republished during the interval between steps 1 and 4), then the CSYNC record obtained in the second set SHOULD NOT be processed (rapidly changing child zones may need special consideration or processing). The operation MAY be restarted or retried in the future.

If the soaminimum flag is set and the SOA serial numbers are equal but less than the CSYNC record's SOA Serial Field [RFC1982], the record MUST NOT be processed. If state is being kept by the parental agent and the SOA serial number is less than the last time a CSYNC record was processed, this CSYNC record SHOULD NOT be processed. Similarly, if state is being kept by the parental agent and the SOA Serial Field of the CSYNC record is less than the SOA Serial Field of the CSYNC record from last time, then this CSYNC record SHOULD NOT be processed.

If a failure occurs of any kind while trying to obtain any of the required data, or if DNSSEC fails to validate all of the data returned for these queries as "secure", then this CSYNC record MUST NOT be processed.

See the "Operational Consideration" section (Section Section 4) for additional guidance about processing.

3.2. CSYNC Record Types

This document defines how the following record types may be processed if the CSYNC Type Bit Map field indicates they are to be processed.

3.2.1. The NS type

The NS type flag indicates that the NS records from the child zone should be copied into the parent's delegation information records for the child.

NS records found within the child's zone should be copied verbatim (with the exception of the TTL field, for which the parent MAY want to select a different value) and the result published within the parent zone should be an exact matching set of NS records. If the child has published a new NS record within their set, this record should be added to the parent zone. Similarly, if NS records in the parent's delegation records for the child contain records that have been removed in the child's NS set, then they should be removed in the parent's set as well.

Parental agents MAY refuse to perform NS updates if the replacement records fail to meet NS record policies required by the parent zone (e.g. "every child zone must have at least 2 NS records"). Parental agents MUST NOT perform NS updates if there are no NS records returned in a query, as verified by DNSSEC denial of existence protection. This situation should never happen unless the child nameservers are misconfigured.

Note that it is permissible for a child's nameserver to return a CSYNC record that removes the queried nameserver itself from the future NS or address set.

3.2.2. The A and AAAA types

The A and AAAA type flags indicates that the A and AAAA address glue records for in-bailiwick NS records within the child zone should be copied verbatim (with the exception of the TTL field, for which the parent MAY want to select a different value) into the parent's delegation information.

Queries should be sent by the parental agent to determine the A and AAAA record addresses for each NS record within a NS set for the child that are in-bailiwick.

Note: only the matching types should be queried. E.g., if the AAAA bit has not been set, then the AAAA records (if any) in the parent's delegation should remain as is. If a given address type is set and the child's zone contains no data for that type (as proven by appropriate NSEC or NSEC3 records), then the result in the parent's delegation records for the child should be an empty set. However, if the end result of processing would leave no glue records present in the parent zone for any of the of the in-bailiwick NS records, then

the parent MUST NOT update the glue address records. I.E., if the result of the processing would leave no in-bailiwick A or AAAA records when there are in-bailiwick NS records, then processing of the address records can not happen as it would leave the parent/child relationship without any address linkage.

The procedure for querying for A and AAAA records MUST occur after the procedure, if required, for querying for NS records as defined in Section Section 3.2.1. This ensures that the right set of NS records is used as provided by the current NS set of the child. I.e., for CSYNC records that have the NS bit set, the NS set used should be the one pulled from the child while processing the CSYNC record. For CSYNC records without the NS bit set, the existing NS records within the parent should be used to determine which A and/or AAAA records to update.

4. Operational Considerations

There are a number of important operational aspects to consider when deploying a CSYNC RRTYPE.

4.1. Error Reporting

There is no inline mechanism for a parental agent to report errors to operators of child zones. Thus, the only error reporting mechanisms must be out of band, such as through a web console or over email. Parental agents should, at a minimum, at least log errors encountered when processing CSYNC records. Child operators utilizing the "immediate" flag that fail to see an update within the parental agent's specified operational window should access the parental agent's error logging interface to determine why an update failed to be processed.

4.2. Child Nameserver Selection

Parental agents will need to poll child nameservers in search of CSYNC records and related data records.

Parental agents MAY perform best-possible verification by querying all NS records for available data to determine which has the most recent SOA and CSYNC version (in an ideal world, they would all be equal, but this is not possible in practice due to synchronization delays and transfer failures).

Parental agents may offer a configuration interface to allow child operators to specify which nameserver should be considered the master to send data queries too. Note that this master could be a different

nameserver than the publically listed nameservers in the NS set (i.e., it may be a "hidden master").

Parental agents with a large number of clients may choose to offer a programmatic interface to let their children indicate that new CSYNC records and data are available for polling rather than polling every child on a frequent basis.

Children that wish to phase out a nameserver will need to publish the CSYNC record to the nameserver being removed and wait for the parental agent to process the published record before turning off the service. This is required because the child can not control which nameserver in the existing NS set the parental agent may choose to query when performing CSYNC processing.

4.3. Out-of-balliwick NS Records

When a zone contains NS records where the domain-name pointed at does not fall within the zone itself, there is no way for the parent to safely update the associated glue records. Thus, the child DNS operator MAY indicate that the NS records should be synchronized, and MAY set any glue record flags (A, AAAA) as well, but the parent will only update those glue records which are below the child's delegation point.

Children deploying NS records pointing to domain-names within their own children (the "grandchildren") SHOULD ensure the grandchildren's associated glue records are properly set before publishing the CSYNC record. I.e., it is imperative that proper communication and synchronization exist between the child and the grandchild.

4.4. Documented Parental Agent Type Support

Parental agents that support processing CSYNC records SHOULD publicly document the following minimum processing characteristics:

- The fact that they support CSYNC processing

- The Type Bit Map bits they support

- The frequency with which they poll clients (which may also be configurable by the client)

- If they support the "immediate" flag

- If they poll a child's single nameserver, a configured list of nameservers, or all of the advertised nameservers when querying records

If they support SOA serial number caching to avoid issues with regression and/or replay

Where errors for CSYNC processing are published

If they support sending queries to a "hidden master".

4.5. Removal of the CSYNC records

Children MAY remove the CSYNC record upon noticing that the parent zone has published the required records, thus eliminating the need for the parent to continually query for the CSYNC record and all corresponding records. By removing the CSYNC record from the child zone, the parental agent will only need to perform the query for the CSYNC record and can stop processing when it finds it missing. This will reduce resource usage by both the child and the parental agent.

4.6. Parent/Child/Grandchild Glue Synchronization

When a child needs to publish a CSYNC record that synchronizes NS and A/AAAA glue records and the NS record is actually pointing to a child of the child (a grandchild of the parent), then it is critical that the glue records in the child point to the proper real addresses records published by the grandchild. It is assumed that if a child is using a grandchild's nameserver that they must be in careful synchronization. Specifically, this specification requires this to be the case.

5. Security Considerations

This specification requires the use of DNSSEC in order to determine that the data being updated was unmodified by third-parties. Parental agents implementing CSYNC processing MUST ensure all DNS transactions are validated by DNSSEC as "secure". Clients deploying CSYNC MUST ensure their zones are signed, current and properly linked to the parent zone with a DS record that points to an appropriate DNSKEY of the child's zone.

This specification does not address how to perform bootstrapping operations to get the required initial DNSSEC-secured operating environment in place. Additionally, this specification was not designed to synchronize DNSSEC security records, such as DS pointers, or the CSYNC record itself. Thus, implementations of this protocol MUST NOT use it to synchronize DS records, DNSKEY materials, CDS records, CDNSKEY records, or CSYNC records. Similarly, future documents extending this protocol MUST NOT offer the ability to synchronize DS, DNSKEY materials, CDS records, CDNSKEY records, or

CSYNC records. For such a solution, please see the complimentary solution [RFC7344] for maintaining security delegation information.

To ensure that an older CSYNC record making use of the soaminimum flag can not be replayed to revert values, the SOA serial number MUST NOT be incremented by more than 2^{16} during the lifetime of the signature window of the associated RRSIGs signing the SOA and CSYNC records. Note that this is independent of whether or not the increment causes the 2^{32} bit serial number field to wrap.

6. IANA Considerations

This document defines a new DNS Resource Record Type, named "CSYNC". The IANA is requested to assign a code point from the "Resource Record (RR) TYPES" sub-registry of the "Domain Name System (DNS) Parameters" registry (<http://www.iana.org/assignments/dns-parameters>) for this record.

TYPE	Value	Meaning	Reference
-----	-----	-----	-----
CSYNC	TBD	Child To Parent Synchronization	[This document]

The IANA is also requested to create and maintain a sub-registry (the "Child Synchronization (CSYNC) Flags" registry) of the "Domain Name System (DNS) Parameters" registry. The initial values for this registry are below.

A "Standards Action" [RFC5226] is required for the assignment of new flag value.

This registry will hold a set of single-bit "Flags" for use in the CSYNC record within the 16 bit Flags field. Thus, a maximum of 16 flags may be defined.

The initial assignments in this registry are:

Bit	Flag	Description	Reference
----	-----	-----	-----
Bit 0	immediate	Immediately process this CSYNC record.	[This document, Section 3]
Bit 1	soaminimum	Require a SOA serial number greater than the one specified.	[This document, Section 2.1.1.1]

For new assignments to be made to this registry, a new standards track RFC must be published via a Standards Action.

7. Acknowledgments

A thank you goes out to Warren Kumari and Olafur Gu[eth]mundsson, who's work on the CDS record type helped inspire the work in this document, as well as the definition for the "parental agent" definition and significant contributions to the text. A thank you also goes out to Ed Lewis, with whom the author held many conversations with about the issues surrounding parent/child relationships and synchronization. Much of the work in this document is derived from the careful existing analysis of these three esteemed colleagues. Thank you to the following people who have contributed text or detailed reviews to the document (in no particular order): Matthijs Mekking, Petr Spacek, 神明達哉, Pete Resnick, Joel Jaeggli, Brian Haberman, Warren Kumari, Adrian Farrel, Alia Atlas, Barry Leiba, Richard Barnes, Stephen Farrell, and Ted Lemon. Lastly, the DNSOP working group chairs Tim Wicinski and Suzanne Woolf have been a tremendous help in getting this draft moving forward to publication.

A special thanks goes to Roy Arends, for taking the "bite out of that hamburger" challenge while discussing this document.

8. References

8.1. Normative References

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.
- [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.

8.2. Informative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, September 2014.

Author's Address

Wes Hardaker
Parsons, Inc.
P.O. Box 382
Davis, CA 95617
US

Phone: +1 530 792 1913
Email: ietf@hardakers.net

dnsop
Internet-Draft
Intended status: Informational
Expires: December 12, 2014

W. Kumari
Google
O. Gudmundsson
Shinkuro Inc.
G. Barwood

June 10, 2014

Automating DNSSEC Delegation Trust Maintenance
draft-ietf-dnsop-delegation-trust-maintenance-14

Abstract

This document describes a method to allow DNS operators to more easily update DNSSEC Key Signing Keys using the DNS as communication channel. The technique described is aimed at delegations in which it is currently hard to move information from the child to parent.

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 12, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	4
1.2. Requirements Notation	4
2. Background	4
2.1. DNS Delegations	4
2.2. Relationship Between Parent and Child DNS Operator	5
2.2.1. Solution Space	6
2.2.2. DNSSEC key change process	6
3. CDS / CDNSKEY (Child DS / Child DNSKEY) Record Definitions	7
3.1. CDS Resource Record Format	8
3.2. CDNSKEY Resource Record Format	8
4. Automating DS Maintenance With CDS / CDNSKEY records	8
4.1. CDS / CDNSKEY Processing Rules	8
5. CDS / CDNSKEY Publication	9
6. Parent Side CDS / CDNSKEY Consumption	9
6.1. Detecting a Changed CDS / CDNSKEY	9
6.1.1. CDS / CDNSKEY Polling	10
6.1.2. Polling Triggers	10
6.2. Using the New CDS / CDNSKEY Records	11
6.2.1. Parent Calculates DS	11
7. IANA Considerations	12
8. Privacy Considerations	12
9. Security Considerations	12
10. Acknowledgements	14
11. References	14
11.1. Normative References	14
11.2. Informative References	15
Appendix A. RRR background	15
Appendix B. CDS key rollover example	16
Appendix C. Changes / Author Notes.	17
Authors' Addresses	21

1. Introduction

The first time a DNS operator signs a zone, they need to communicate the keying material to their parent through some out-of-band method to complete the chain of trust. Depending on the desires of the parent, the child might send their DNSKEY record, a DS record, or both.

Each time the child changes the key that is represented in the parent, the updated and / or deleted key information has to be communicated to the parent and published in the parent's zone. How

this information is sent to the parent depends on the relationship the child has with the parent. In many cases this is a manual process, and not an easy one. For each key change, there may be up to two interactions with the parent. Any manual process is susceptible to mistakes and / or errors. In addition, due to the annoyance factor of the process, operators may avoid changing keys or skip needed steps to publish the new DS at the parent.

DNSSEC provides data integrity to information published in DNS; thus DNS publication can be used to automate maintenance of delegation information. This document describes a method to automate publication of subsequent DS records, after the initial one has been published.

Readers are expected to be familiar with DNSSEC, including [RFC4033], [RFC4034], [RFC4035], [RFC5011] and [RFC6781].

This document outlines a technique in which the parent periodically (or upon request) polls its signed children and automatically publishes new DS records. To a large extent, the procedures this document follows are as described in [RFC6781] section 4.1.2.

This technique is designed to be friendly both to fully automated tools and humans. Fully automated tools can perform all the actions needed without human intervention, and thus can monitor when it is safe to move to the next step.

The solution described in this document only allows transferring information about DNSSEC keys (DS and DNSKEY) from the child to the parental agent. It lists exactly what the parent should publish, and allows for publication of stand-by keys. A different protocol, [I-D.csync], can be used to maintain other important delegation information, such as NS and glue. These two protocols have been kept as separate solutions because the problems are fundamentally different, and a combined solution is overly complex.

This document describes a method for automating maintenance of the delegation trust information, and proposes a polled / periodic trigger for simplicity. Some users may prefer a different trigger, for example a button on a webpage, a REST interface or a DNS NOTIFY. These alternate / additional triggers are not discussed in this document.

This proposal does not include all operations needed for the maintenance of DNSSEC key material, specifically the initial introduction or complete removal of all keys. Because of this, alternate communications mechanisms must always exist, potentially introducing more complexity.

1.1. Terminology

The terminology we use is defined in this section.

Highlighted roles:

- o Child: "The entity on record that has the delegation of the domain from the parent"
- o Parent: "The domain in which the child is registered"
- o Child DNS Operator: "The entity that maintains and publishes the zone information for the child DNS"
- o Parental Agent: "The entity that the child has relationship with, to change its delegation information"
- o Provisioning system: "A system that the operator of the master DNS server operates to maintain the information published in the DNS. This includes the systems that sign the DNS data"

1.2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Background

2.1. DNS Delegations

DNS operation consists of delegations of authority. For each delegation there are (most of the time) two parties: the parent and the child.

The parent publishes information about the delegations to the child; for the name servers it publishes an NS [RFC1035] RRset that lists a hint for name servers that are authoritative for the child. The child also publishes a NS RRset, and this set is the authoritative list of name servers to the child zone.

The second RRset the parent sometimes publishes is the DS [RFC4034] set. The DS RRset provides information about the DNSKEY(s) that the child has told the parent it will use to sign its DNSKEY RRset. In DNSSEC trust relationship between zones is provided by the following chain:

parent DNSKEY --> DS --> child DNSKEY.

A prior proposal [I-D.auto-cpsync] suggested that the child send an "update" to the parent via a mechanism similar to Dynamic Update. The main issue became: How does the child find the actual parental agent/server to send the update to? While that could have been solved via technical means, it failed to reach consensus. There is also a similar proposal in [I-D.parent-zones].

As the DS record can only be present at the parent ([RFC4034]), some other method is needed to automate which DNSKEYs are picked to be represented in the parent zone's DS records. One possibility is to use flags in the DNSKEY record. If the SEP bit is set, this indicates that the DNSKEY is intended for use as a secure entry point. This DNSKEY signs the DNSKEY RRset, and the Parental Agent can calculate DS records based on that. But this fails to meet some operating needs, including the child having no influence what DS digest algorithms are used and DS records can only be published for keys that are in the DNSKEY RRset, and thus this technique would not be compatible with Double-DS ([RFC6781]) key rollover.

2.2. Relationship Between Parent and Child DNS Operator

In practical application, there are many different relationships between the parent and Child DNS Operators. The type of relationship affects how the Child DNS Operator communicates with the parent. This section will highlight some of the different situations, but is by no means a complete list.

Different communication paths:

- o Direct/API: The child can change the delegation information via automated/scripted means. EPP[RFC5730], used by many TLDs is an example of this. Other examples are web-based programmatic interfaces that Registrars make available to their Resellers.
- o User Interface: The Child uses a (web) site set up by the Parental Agent for updating delegation information.
- o Indirect: The communication has to be transmitted via out-of-band between two parties, such as by email or telephone. This is common when the Child's DNS operator is neither the child itself nor the Registrar for the domain but a third party.
- o Multi-step Combinations: The information flows through an intermediary. It is possible, but unlikely, that all the steps are automated via API's and there are no humans involved.

A domain name holder (Child) may operate its own DNS servers or outsource the operation. While we use the word parent as a singular, parent can consist of single entity or a composite of many discrete parts that have rules and roles. We refer to the entity that the child corresponds with as the Parent.

An organization (such as an enterprise) may delegate parts of its name-space to be operated by a group that is not the same as that which operates the organization's DNS servers. In some of these cases the flow of information is handled in either an ad hoc manner or via some corporate mechanism; this can range from email to fully-automated operation.

2.2.1. Solution Space

This document is aimed at the cases in which there is a separation between the child and parent.

A further complication is when the Child DNS Operator is not the Child. There are two common cases of this:

- a) The Parental Agent (e.g. registrar) handles the DNS operation.
- b) A third party takes care of the DNS operation.

If the Parental Agent is the DNS operator, life is much easier; the Parental Agent can inject any delegation changes directly into the Parent's Provisioning system. The techniques described below are not needed in the case when Parental Agent is the DNS operator.

In the case of a third party DNS operator, the Child either needs to relay changes in DNS delegation or give the Child DNS Operator access to its delegation/registration account.

Some parents want the child to express their DNSKEYs in the form of DS records, while others want to receive the DNSKEY records and calculate the DS records themselves. There is no consensus on which method is better; both have good reasons to exist. This solution is DS vs DNSKEY agnostic, and allows operation with either.

2.2.2. DNSSEC key change process

After a Child DNS Operator first signs the zone, there is a need to interact with the Parent, for example via a delegation account interface, to "upload / paste-in the zone's DS information". This action of logging in through the delegation account user interface authenticates that the user is authorized to change delegation information for the child published in the parent zone. In the case

where the Child DNS Operator does not have access to the registration account, the Child needs to perform the action.

At a later date, the Child DNS Operator may want to publish a new DS record in the parent, either because they are changing keys, or because they want to publish a stand-by key. This involves performing the same process as before. Furthermore when this is a manual process with cut and paste, operational mistakes will happen -- or worse, the update action is not performed at all.

The Child DNS Operator may also introduce new keys, and can do so when old keys exist and can be used. The Child may also remove old keys, but this document does not support removing all keys. This is to avoid making signed zones unsigned. The Child may not enroll the initial key or introduce a new key when there are no old keys that can be used (without some additional, out of band, validation of the keys), because there is no way to validate the information.

3. CDS / CDNSKEY (Child DS / Child DNSKEY) Record Definitions

This document specifies two new DNS resource records, CDS and CDNSKEY. These records are used to convey, from one zone to its parent, the desired contents of the zone's DS resource record set residing in the parent zone.

The CDS / CDNSKEY resource records are published in the child zone and gives the child control of what is published for it in the parental zone. The child can publish these manually, or they can be automatically maintained by DNS provisioning tools. The CDS / CDNSKEY RRset expresses what the child would like the DS RRset to look like after the change; it is a "replace" operation, and it is up to the software that consumes the records to translate that into the appropriate add/delete operations in the provisioning systems (and in the case of CDNSKEY, to generate the DS from the DNSKEY). If no CDS / CDNSKEY RRset is present in child, this means that no change is needed.

[[RFC Editor: Please remove this paragraph before publication]
Version -04 of the ID that became this working group document (<http://tools.ietf.org/id/draft-kumari-ogud-dnsop-cds-04.txt>) defined a new record (CTA) that could hold either a DS or a DNSKEY record (with a selector to differentiate between them). In a shocking development, there was almost full consensus that this was horrid :-)
]

3.1. CDS Resource Record Format

The wire and presentation format of the CDS ("Child DS") resource record is identical to the DS record [RFC4034]. IANA has allocated RR code 59 for the CDS resource record via expert review [I-D.ds-publish]. The CDS RR uses the same registries as DS for its fields.

No special processing is performed by authoritative servers or by resolvers, when serving or resolving. For all practical purposes CDS is a regular RR type.

3.2. CDNSKEY Resource Record Format

The wire and presentation format of the CDNSKEY ("Child DNSKEY") resource record is identical to the DNSKEY record. IANA has allocated RR code TBA1 for the CDNSKEY resource record via expert review. The CDNSKEY RR uses the same registries as DNSKEY for its fields.

No special processing is performed by authoritative servers or by resolvers, when serving or resolving. For all practical purposes CDNSKEY is a regular RR type.

4. Automating DS Maintenance With CDS / CDNSKEY records

CDS / CDNSKEY resource records are intended to be "consumed" by delegation trust maintainers. The use of CDS / CDNSKEY is OPTIONAL.

If the child publishes either the CDS or the CDNSKEY resource record, it SHOULD publish both. If the child knows which the parent consumes, it MAY choose to only publish that record type (for example, some children wish the parent to publish a DS, but they wish to keep the DNSKEY "hidden" until needed). If the child publishes both, the two RRsets MUST match in content.

4.1. CDS / CDNSKEY Processing Rules

If there are no CDS / CDNSKEY RRset in the child, this signals that no change should be made to the current DS set. This means that, once the child and parent are in sync, the Child DNS Operator MAY remove all CDS and CDNSKEY resource records from the zone. The Child DNS Operator may choose to do this to decrease the size of the zone, or to decrease the workload for the parent (if the parent receives no CDS / CDNSKEY records it can go back to sleep). If it does receive a CDS or CDNSKEY RRset it needs to check them against what is currently published - see Section 5.

Following acceptance rules are placed on the CDS / CDNSKEY resource records as follows:

- o Location: the CDS / CDNSKEY resource records MUST be at the child zone apex.
- o Signer: MUST be signed with a key that is represented in both the current DNSKEY and DS RRsets (unless the parent uses the CDS / CDNSKEY RRset for initial enrollment, in that case the parent validates the CDS / CDNSKEY through some other means (see Section 6.1 and the Security Considerations.)
- o Continuity: MUST NOT break the current delegation if applied to DS RRset.

If any these conditions fail the CDS / CDNSKEY resource record MUST be ignored, and this error SHOULD be logged.

5. CDS / CDNSKEY Publication

The Child DNS Operator publishes CDS and / or CDNSKEY resource records. In order to be valid, the CDS / CDNSKEY RRset MUST be compliant with the rules in Section 4.1. When the Parent DS is "in sync" with the CDS / CDNSKEY resource records, the Child DNS Operator MAY delete the CDS / CDNSKEY record(s); the child can determine if this is the case by querying for DS records in the parent

6. Parent Side CDS / CDNSKEY Consumption

The CDS / CDNSKEY RRset SHOULD be used by the Parental Agent to update the DS RRset in the parent zone. The Parental Agent for this uses a tool that understands the CDS / CDNSKEY signing rules from Section 4.1 so it might not be able to use a standard validator.

The parent MUST choose to use either CDNSKEY or CDS resource records as their default updating mechanism. The parent MAY only accept either CDNSKEY or CDS, but it MAY also accept both, so it can use the other in the absence of the default updating mechanism, but it MUST NOT expect there to be both.

6.1. Detecting a Changed CDS / CDNSKEY

How the Parental Agent gets the CDS / CDNSKEY RRset may differ, below are two examples as how this can take place.

Polling The Parental Agent operates a tool that periodically checks each of the children that has a DS record to see if there is a CDS or CDNSKEY RRset.

Pushing The delegation user interface has a button {Fetch DS} when pushed performs the CDS / CDNSKEY processing. If the Parent zone does not contain DS for this delegation then the "push" SHOULD be ignored. If the Parental Agent displays the contents of the CDS / CDNSKEY to the user and gets confirmation that this represents their key, the Parental Agent MAY use this for initial enrollment (when the Parent zone does not contain the DS for this delegation).

In either case the Parental Agent MAY apply additional rules that defer the acceptance of a CDS / CDNSKEY change, these rules may include a condition that the CDS / CDNSKEY remains in place and valid for some time period before it is accepted. It may be appropriate in the "Pushing" case to assume that the Child is ready and thus accept changes without delay.

6.1.1. CDS / CDNSKEY Polling

This is the only defined use of CDS / CDNSKEY resource records in this document. There are limits to the scalability of polling techniques, thus some other mechanism is likely to be specified later that addresses CDS / CDNSKEY resource record usage in the situation where polling does not scale to. Having said that, Polling will work in many important cases such as enterprises, universities and smaller TLDs. In many regulatory environments the registry is prohibited from talking to the registrant. In most of these cases the registrant has a business relationship with the registrar, and so the registrar can offer this as a service.

If the CDS / CDNSKEY RRset does not exist, the Parental Agent MUST take no action. Specifically it MUST NOT delete or alter the existing DS RRset.

6.1.2. Polling Triggers

It is assumed that other mechanisms will be implemented to trigger the parent to look for an updated CDS / CDNSKEY RRsets. As the CDS / CDNSKEY resource records are validated with DNSSEC, these mechanisms can be unauthenticated. As an example, a child could telephone its parent and request that they process the new CDS or CDNSKEY resource records or an unauthenticated POST could be made to a webserver (with rate-limiting).

Other documents can specify the trigger conditions.

6.2. Using the New CDS / CDNSKEY Records

Regardless of how the Parental Agent detected changes to a CDS / CDNSKEY RRset, the Parental Agent SHOULD use a DNSSEC validator to obtain a validated CDS / CDNSKEY RRset from the Child zone. A NOT RECOMMENDED exception to this is if the parent performs some additional validation on the data to confirm that it is the "correct" key.

The Parental Agent MUST ensure that previous versions of the CDS / CDNSKEY RRset do not overwrite more recent versions. This MAY be accomplished by checking that the signature inception in the RRSIG for CDS / CDNSKEY RRset is later and / or the serial number on the child's SOA is greater. This may require the Parental Agent to maintain some state information.

The Parental Agent MAY take extra security measures. For example, to mitigate the possibility that a Child's key signing key has been compromised, the Parental Agent may, for example, inform (by email or other methods) the Child DNS Operator of the change. However the precise out-of-band measures that a parent zone takes are outside the scope of this document.

Once the Parental Agent has obtained a valid CDS / CDNSKEY RRset it MUST check the publication rules from section 4.1. In particular the Parental Agent MUST check the Continuity rule and do its best not to invalidate the Child zone. Once checked and if the information in the CDS / CDNSKEY and DS differ it may apply the changes to the parent zone. If the parent consumes CDNSKEY, the parent should calculate the DS before doing this comparison.

6.2.1. Parent Calculates DS

There are cases where the Parent wants to calculate the DS record due to policy reasons. In this case, the Child publishes CDNSKEY records and the parent calculates the DS records on behalf of the children.

When a Parent operates in "calculate DS" mode it can operate in one of two sub-modes:

full: it only publishes DS records it calculates from DNSKEY records,

augment: it will make sure there are DS records for the digest algorithm(s) it requires(s).

In the case where the parent fetches the CDNSKEY RRset and calculates the DS the resulting DS can differ from the CDS published by the

child. It is expected that the differences are only due different set of digest algorithms used.

7. IANA Considerations

IANA has assigned RR Type code 59 for the CDS resource record. This was done for an earlier version of this document[I-D.ds-publish] This document is to become the reference for CDS RRtype.

IANA is requested to assign an RR Type for the CDNSKEY, see below

Type: CDNSKEY

Value: TBD1 (60 suggested)

Meaning: DNSKEY(s) the child wants reflected in DS

Reference: This document

8. Privacy Considerations

All of the information handled / transmitted by this protocol is public information published in the DNS.

9. Security Considerations

This work is for the normal case; when things go wrong there is only so much that automation can fix.

If child breaks DNSSEC validation by removing all the DNSKEYs that are represented in the DS set its only repair actions are to contact the parent or restore the DNSKEYs in the DS set.

In the event of a compromise of the server or system generating signatures for a zone, an attacker might be able to generate and publish new CDS / CDNSKEY resource records. The modified CDS / CDNSKEY records will be picked up by this technique and so may allow the attacker to extend the effective time of his attack. If there is a delay in accepting changes to DS, as in [RFC5011], then the attacker needs to hope his activity is not detected before the DS in the parent is changed. If this type of change takes place, the child needs to contact the parent (possibly via a registrar web interface) and remove any compromised DS keys.

A compromise of the account with the parent (e.g. registrar) will not be mitigated by this technique, as the "new registrant" can delete / modify the DS records at will.

While it may be tempting, this SHOULD NOT be used for initial enrollment of keys since there is no way to ensure that the initial key is the correct one. If is used, strict rules for inclusion of keys such as hold down times, challenge data inclusion or similar, MUST be used, along with some kind of challenge mechanism. A child cannot use this mechanism to go from signed to unsigned (publishing an empty CDS / CDNSKEY RRset means no-change should be made in the parent).

The CDS RR type should allow for enhanced security by simplifying process. Since key change is automated, updating a DS RRset by other means may be regarded as unusual and subject to extra security checks.

As this introduces a new mechanism to update information in the parent it MUST be clear who is fetching the records and creating the appropriate records in the parent zone. Specifically some operations may use other mechanisms than what is described here. For example, a registrar may assume that it is maintaining the DNSSEC key information in the registry, and may have this cached. If the registry is fetching the CDS / CDNSKEY RRset then the registry and registrar may have different views of the DNSSEC key material and the result of such a situation is unclear. Therefore, this mechanism SHOULD NOT be use to bypass intermediaries that might cache information and because of that get the wrong state.

If there is a failure in applying changes in the child zone to all DNS servers listed in either parent or child NS set it is possible that the Parental agent may get confused, either because it gets different answers on different checks or CDS RR validation fails. In the worst case, the Parental Agent performs an action reversing a prior action but after the child signing system decides to take the next step in the key change process, resulting in a broken delegation.

DNS is a loosely coherent distributed database with local caching; therefore, it is important to allow old information to expire from caches before deleting DS or DNSKEY records. Similarly, it is important to allow new records to propagate through the DNS before use, see [RFC6781].

It is common practice for users to outsource their DNS hosting to a third-party DNS provider. In order for that provider to be able to maintain the DNSSEC information some users give the provider their registrar login credentials (which obviously has negative security implications). Deploying the solution described in this document allows the 3rd party DNS provider to maintain the DNSSEC information

without giving them the registrar credentials, thereby improving security.

By automating the maintenance of the DNSSEC key information (and removing humans from the process), we expect to decrease the number of DNSSEC related outages, which should increase DNSSEC deployment.

10. Acknowledgements

We would like to thank a large number of folk, including: Mark Andrews, Joe Abley, Jaap Akkerhuis, Roy Arends, Doug Barton, Brian Dickson, Paul Ebersman, Tony Finch, Jim Galvin, Paul Hoffman, Samir Hussain, Tatuya Jinmei, Olaf Kolkman, Stephan Lagerholm, Cricket Liu, Matt Larson, Marco Sanz, Antoin Verschuren, Suzanne Woolf, Paul Wouters, John Dickinson, Timothe Litt and Edward Lewis.

Special thanks to Wes Hardaker for contributing significant text and creating the complementary (CSYNC) solution, and to Patrik Faltstrom, Paul Hoffman, Matthijs Mekking, Mukund Sivaraman and Jeremy C. Reed for text and in-depth review. Brian Carpender provided a good Gen-Art review.

There were a number of other folk with whom we discussed this, apologies for not remembering everyone.

11. References

11.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, September 2007.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, December 2012.

11.2. Informative References

- [I-D.auto-cpsync]
Mekking, W., "Automated (DNSSEC) Child Parent Synchronization using DNS UPDATE", draft-mekking-dnsop-auto-cpsync-01 (work in progress), December 2010.
- [I-D.csync]
Hardaker, W., "Child To Parent Synchronization in DNS", draft-hardaker-dnsop-csync-02 (work in progress), July 2013.
- [I-D.ds-publish]
Barwood, G., "DNS Transport", draft-barwood-dnsop-ds-publish-02 (work in progress), June 2011.
- [I-D.parent-zones]
Andrews, M., "Updating Parent Zones", draft-andrews-dnsop-update-parent-zones-04 (work in progress), November 2013.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, May 2010.

Appendix A. RRR background

RRR is our shorthand for Registry/Registrar/Registrant model of parent child relationship.

In the RRR world, the different parties are frequently from different organizations. In the single enterprise world there are also organizational / geographical / cultural separations that affect how information flows from a Child to the parent.

Due to the complexity of the different roles and interconnections, automation of delegation information has not yet occurred. There have been proposals to automate this, in order to improve the

reliability of the DNS. These proposals have not gained enough traction to become standards.

For example in many of the TLD cases there is the RRR model (Registry, Registrar and Registrant). The Registry operates DNS for the TLD, the Registrars accept registrations and place information into the Registries database. The Registrant only communicates with the Registrar; frequently the Registry is not allowed to communicate with the Registrant. In that case as far as the registrant is concerned the Registrar is the same entity as the Parent.

In many RRR cases the Registrar and Registry communicate via EPP[RFC5730] and use the EPP DNSSEC extension [RFC5910]. In a number of ccTLDs there are other mechanisms in use as well as EPP, but in general there seems to be a movement towards EPP usage when DNSSEC is enabled in the TLD.

Appendix B. CDS key rollover example

This section shows an example on how CDS is used when performing a KSK rollover, this example will demonstrate the the double DS rollover method from section 4.1.2 in [RFC6781]. Other rollovers using CDNSKEY and double KSK are left as an exercise to the reader. The table below does not reflect the ZSK keys they just do not matter during KSK rollovers. The wait states below highlight what RRset needs to expire from caches before progressing to the next step.

Step	State	Parent DS	Child KSK	DNSKEY and CDS signer	Child CDS
	Beginning	A	A	A	
1	Add CDS	A	A	A	AB
Wait	for DS change	A	A	A	AB
2	Updated DS	AB	A	A	AB
Wait	> DS TTL	AB	A	A	AB
3	Actual Rollover	AB	B	B	AB
Wait	> DNSKEY TTL	AB	B	B	AB
4	Child Cleanup	AB	B	B	B
5	Parent cleans	B	B	B	B
6	Optional CDS delete	B	B	B	

Table 1: States

Appendix C. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

WG-13 to WG-14 IETF Last call and IESG processing

- o Applied text fixes from Phil Pennock
- o Addressed comments from Brian Carpenter GEN-ART review.
- o Barry Leiba wanted better IANA considerations and suggested some text changes in 6.1 and 6.2.1
- o Reformatted the Appendix B table to be clearer.

WG-12 to WG-13

- o Added appendix B showing Key rollover using CDS.

WG-11 to WG-12

- o Many nits and helpful grammar fixes from Jeremy C. Reed.

WG-10 to WG-11

- o More useful text from Matthijs.
- o Explained why the child might want to remove the CDS / CDNSKEY Records.

WG-09 to WG-10

- o Incorporated off list comments from Stephan Lagerholm. Largest change is fixing discrepancy between parent MAY perform OOB validation and the Signer rule in 4.1. Clarified by updating signer rule to allow enrolment if validation is performed OOB.

WG-08 to WG-09

- o New text from Paul Hoffman for the first paragraph of the intro.
- o And a modification from Jaap.

WG-07 to WG-08

- o Incorporated text from Antoin Verschuren at the end of Section 6.
- o Comments from Paul Hoffman and Tim W

WG-06 to WG-07

- o Incorporated nits / editorial comments from Tim Wicinski.
- o
 - * Reference for Mark's draft was incorrect, Wes Hardaker doesn't work for ISC :-P
 - * Normalized CDS record / CDS resource record / records / etc.
 - * Language cleanup / nits / poor grammar.
 - * removed "punted" colloquialism.

WG-05 to WG-06

- o Consensus (according to me!) was that mail thread said "Child MAY remove the CDS record". Changed to accommodate.
- o "The proposal below can operate with both models, but the child needs to be aware of the parental policies." - removed "but the child needs to be aware of the parental policies.". This is no longer true, as we suggest publishing both CDS and CDSNKEY.
- o Added: "without some additional out of band process" to "The Child may not enroll the initial key or introduce a new key when there are no old keys that can be used (without some additional, out of band, validation of the keys), because there is no way to validate the information."
- o Added a bit to the IANA section, requesting that TBA1 be replaced with the IANA allocated code.
- o Removed: "Some parents prefer to accept DNSSEC key information in DS format, some parents prefer to accept it in DNSKEY format, and calculate the DS record on the child's behalf. Each method has pros and cons, both technical and policy. This solution is DS vs DNSKEY agnostic, and allows operation with either." from Section 4 as it is covered in Section 2.2.1
- o Remove a bunch of comments from the XML. I was getting tired of scrolling past them. If the authors need them back, they are in SVN commit 47.

WG-04 to WG-05

- o More comments from Patrik, Paul and Ed.

- o Many nits and fixes from Matthijs Mekking.
- o Outstanding question: Should we remove the "Child SHOULD remove the CDS record" text? Mail sent to list.

WG-03 to WG-04

- o Large number of comments and changes from Patrik.

WG-02 to WG-03

- o Fixed some references to RFC 5011 - from Samir Hussain.
- o Fixed some spelling / typos - from Samir Hussain.
- o A number of clarifications on the meaning on an empty / non-existent CDS RRset - thanks to JINMEI, Tatuya
- o Be consistent in mentioning both CDS and CDNSKEY throughout the document.

WG-01 to WG-02

- o Many nits and suggestions from Mukund.
- o Matthijs: " I still think that it is too strong that the Child DNS Operator SHOULD/MUST delete the CDS RRset when the Parent DS is "in sync". This should be a MAY"

WG-00 to WG-01

- o Addressed Vancouver: "Paul Hoffmann: NOT ready for WGLC. None of the 2 documents explain why there is a split between the two strategies." Thanks to Paul for providing text.

From -05 to WG-00:

- o Nothing changed, resubmit under new name.

From 04 to 05

- o Renamed the record back to CDS.

From 03 to 04.

- o Added text explaining that CDS and CSYNC complement each other, not replace or compete.

- o Changed format of record to be <selector> <data> to allow the publication of DS **or** DNSKEY.
- o Bunch of text changed to cover the above.
- o Added a bit more text on the polling scaling stuff, expectation that other triggers will be documented.

From 02 to 03

- o Applied comments by Matthijs Mekking
- o Incorporated suggestions from Edward Lewis about structure
- o Reworked structure to be easier for implementors to follow
- o Applied many suggestions from a wonderful thorough review by John Dickinson
- o Removed the going Unsigned option

From 01 to 02

- o Major restructuring to facilitate easier discussion
- o Lots of comments from DNSOP mailing list incorporated, including making draft DNSKEY/DS neutral, explain different relationships that exists,
- o added more people to acks.
- o added description of enterprise situations
- o Unified on using Parental Agent over Parental Representative
- o Removed redundant text when possible
- o Added text to explain what can go wrong if not all child DNS servers are in sync.
- o Reference prior work by Matthijs Mekking
- o Added text when parent calculates DS from DNSKEY

From - to -1.

- o Removed from section .1: "If a child zone has gone unsigned, i.e. no DNSKEY and no RRSIG in the zone, the parental representative

MAY treat that as intent to go unsigned. (NEEDS DISCUSSION)."
Added new text at end. -- suggestion by Scott Rose 20/Feb/13.

- o Added some background on the different DNS Delegation operating situations and how they affect interaction of parties. This moved some blocks of text from later sections into here.
- o Number of textual improvements from Stephan Lagerholm
- o Added motivation why CDS is needed in CDS definition section
- o Unified terminology in the document.
- o Much more background

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Olafur Gudmundsson
Shinkuro Inc.
4922 Fairmont Av, Suite 250
Bethesda, MD 20814
USA

Email: ogud@ogud.com

George Barwood
33 Sandpiper Close
Gloucester GL2 4LZ
United Kingdom

Email: george.barwood@blueyonder.co.uk

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 16, 2015

S. Morris
ISC
J. Ihren
Netnod
J. Dickinson
Sinodun
W. Mekking
NLnet Labs
October 13, 2014

DNSSEC Key Rollover Timing Considerations
draft-ietf-dnsop-dnssec-key-timing-06.txt

Abstract

This document describes the issues surrounding the timing of events in the rolling of a key in a DNSSEC-secured zone. It presents timelines for the key rollover and explicitly identifies the relationships between the various parameters affecting the process.

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 April 16, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Key Rolling Considerations	3
1.2. Types of Keys	4
1.3. Terminology	4
1.4. Limitation of Scope	4
2. Rollover Methods	5
2.1. ZSK Rollovers	5
2.2. KSK Rollovers	6
3. Key Rollover Timelines	8
3.1. Key States	8
3.2. Zone-Signing Key Timelines	9
3.2.1. Pre-Publication Method	9
3.2.2. Double-Signature Method	12
3.3. Key-Signing Key Rollover Timelines	14
3.3.1. Double-KSK Method	14
3.3.2. Double-DS Method	16
3.3.3. Double-RRset Method	19
3.3.4. Interaction with Configured Trust Anchors	21
3.3.5. Introduction of First Keys	22
4. Standby Keys	23
5. Algorithm Considerations	24
6. Summary	24
7. IANA Considerations	25
8. Security Considerations	25
9. Acknowledgements	25
10. Normative References	25
Appendix A. List of Symbols	25
Appendix B. Change History (To be removed on publication)	28
Authors' Addresses	30

1. Introduction

1.1. Key Rolling Considerations

When a zone is secured with DNSSEC, the zone manager must be prepared to replace ("roll") the keys used in the signing process. The rolling of keys may be caused by compromise of one or more of the existing keys, or it may be due to a management policy that demands periodic key replacement for security or operational reasons. In order to implement a key rollover, the keys need to be introduced into and removed from the zone at the appropriate times. Considerations that must be taken into account are:

- o DNSKEY records and associated information (such as the DS records or RRSIG records created with the key) are not only held at the authoritative nameserver, they are also cached by resolvers. The data on these systems can be interlinked, e.g., a validating resolver may try to validate a signature retrieved from a cache with a key obtained separately.
- o Zone "boot-strapping" events, where a zone is signed for the first time, can be common in configurations where a large number of zones are being served. Procedures should be able to cope with the introduction of keys into the zone for the first time as well as "steady-state", where the records are being replaced as part of normal zone maintenance.
- o To allow for an emergency re-signing of the zone as soon as possible after a key compromise has been detected, standby keys (additional keys over and above those used to sign the zone) need to be present.
- o A query for the DNSKEY RRset returns all DNSKEY records in the zone. As there is limited space in the UDP packet (even with EDNS0 support), key records no longer needed must be periodically removed. (For the same reason, the number of standby keys in the zone should be restricted to the minimum required to support the key management policy.)

Management policy, e.g., how long a key is used for, also needs to be considered. However, the point of key management logic is not to ensure that a rollover is completed at a certain time but rather to ensure that no changes are made to the state of keys published in the zone until it is "safe" to do so ("safe" in this context meaning that at no time during the rollover process does any part of the zone ever go bogus). In other words, although key management logic enforces policy, it may not enforce it strictly.

A high-level overview of key rollover can be found in [RFC6781]. In contrast, this document focuses on the low-level timing detail of two classes of operations described there, the rollover of zone-signing keys (ZSKs), and the rollover of key-signing keys (KSKs).

Note that the process for the introduction of keys into a zone is different to that of rolling a key; see Section 3.3.5 for more information about that topic.

1.2. Types of Keys

Although DNSSEC validation treats all keys equally, [RFC4033] recognises the broad classification of ZSKs and KSKs. A ZSK is used to authenticate information within the zone; a KSK is used to authenticate the zone's DNSKEY RRset. The main implication for this distinction concerns the consistency of information during a rollover.

During operation, a validating resolver must use separate pieces of information to perform an authentication. At the time of authentication, each piece of information may be in its cache or may need to be retrieved from an authoritative server. The rollover process needs to happen in such a way that at all times during the rollover the information is consistent. With a ZSK, the information is the RRSIG (plus associated RRset) and the DNSKEY. These are both obtained from the same zone. In the case of the KSK, the information is the DNSKEY and DS RRset with the latter being obtained from a different zone.

Although there are similarities in the algorithms to roll ZSKs and KSKs, there are a number of differences. For this reason, the two types of rollovers are described separately.

1.3. Terminology

The terminology used in this document is as defined in [RFC4033] and [RFC5011].

A number of symbols are used to identify times, intervals, etc. All are listed in Appendix A.

1.4. Limitation of Scope

This document represents current thinking at the time of publication. However, the subject matter is evolving and it is not possible for the document to be comprehensive. In particular, it does not cover:

- o Rolling a key that is used as both a ZSK and KSK.
- o Algorithm rollovers. Only the rolling of keys of the same algorithm are described here, not transitions between algorithms.

The reader is therefore reminded that DNSSEC is, as of date of publication, in the early stages of deployment, and best practices may further develop over time.

2. Rollover Methods

2.1. ZSK Rollovers

For ZSKs, the issue for the zone operator/signer is to ensure that any caching validator which has access to a particular signature also has access to the corresponding valid ZSK.

A ZSK can be rolled in one of three ways:

- o Pre-Publication: described in [RFC6781], the new key is introduced into the DNSKEY RRset which is then re-signed. This state of affairs remains in place for long enough to ensure that any cached DNSKEY RRsets contain both keys. At that point signatures created with the old key can be replaced by those created with the new key, and the old signatures removed. During the re-signing process (which may or may not be atomic depending on how the zone is managed), it doesn't matter which key an RRSIG record retrieved by a resolver was created with; cached copies of the DNSKEY RRset will contain both the old and new keys.

Once the zone contains only signatures created with the new key, there is an interval during which RRSIG records created with the old key expire from caches. After this, there will be no signatures anywhere that were created using the old key, and it can be removed from the DNSKEY RRset.

- o Double-Signature: also mentioned in [RFC6781], this involves introducing the new key into the zone and using it to create additional RRSIG records; the old key and existing RRSIG records are retained. During the period in which the zone is being signed (again, the signing process may not be atomic), validating resolvers are always able to validate RRSIGs: any combination of old and new DNSKEY RRset and RRSIGs allows at least one signature to be validated.

Once the signing process is complete and enough time has elapsed to make sure that all validators that have the DNSKEY and

signatures in cache have both the old and new information, the old key and signatures can be removed from the zone. As before, during this period any combination of DNSKEY RRset and RRSIGs will allow validation of at least one signature.

- o Double-RRSIG: strictly speaking, the use of the term "Double-Signature" above is a misnomer as the method is not only double signature, it is also double key as well. A true Double-Signature method (here called the Double-RRSIG method) involves introducing new signatures in the zone (while still retaining the old ones) but not introducing the new key.

Once the signing process is complete and enough time has elapsed to ensure that all caches that may contain an RR and associated RRSIG have a copy of both signatures, the key is changed. After a further interval during which the old DNSKEY RRset expires from caches, the old signatures are removed from the zone.

Of the three methods, Double-Signature is conceptually the simplest - introduce the new key and new signatures, then approximately one TTL later remove the old key and old signatures. It is also the fastest, but suffers from increasing the size of the zone and the size of responses.

Pre-Publication is more complex - introduce the new key, approximately one TTL later sign the records, and approximately one TTL after that remove the old key. It does however keep the zone and response sizes to a minimum.

Double-RRSIG is essentially the reverse of Pre-Publication - introduce the new signatures, approximately one TTL later change the key, and approximately one TTL after that remove the old signatures. However, it has the disadvantage of the Pre-Publication method in terms of time taken to perform the rollover, the disadvantage of the Double-Signature rollover in terms of zone and response sizes, and none of the advantages of either. For these reasons, it is unlikely to be used in any real-world situations and so will not be considered further in this document.

2.2. KSK Rollovers

In the KSK case, there should not be a problem that a caching validator does not have access to a particular signature that corresponds to a valid KSK. The KSK is only used for one signature (that over the DNSKEY RRset) and both the key and the signature travel together. Instead, the issue is to ensure that the KSK is trusted.

Trust in the KSK is either due to the existence of a signed and validated DS record in the parent zone or an explicitly configured trust anchor. If the former, the rollover algorithm will need to involve the parent zone in the addition and removal of DS records, so timings are not wholly under the control of the zone manager. If the latter, [RFC5011] timings will be needed to roll the keys. (Even in the case where authentication is via a DS record, the zone manager may elect to include [RFC5011] timings in the key rolling process so as to cope with the possibility that the key has also been explicitly configured as a trust anchor.)

It is important to note that the need to interact with the parent does not preclude the development of key rollover logic; in accordance with the goal of the rollover logic being able to determine when a state change is "safe", the only effect of being dependent on the parent is that there may be a period of waiting for the parent to respond in addition to any delay the key rollover logic requires. Although this introduces additional delays, even with a parent that is less than ideally responsive the only effect will be a slowdown in the rollover state transitions. This may cause a policy violation, but will not cause any operational problems.

Like the ZSK case, there are three methods for rolling a KSK:

- o Double-KSK: the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key. After waiting for the old RRset to expire from caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset.
- o Double-DS: the new DS record is published. After waiting for this change to propagate into caches, the KSK is changed. After a further interval during which the old DNSKEY RRset expires from caches, the old DS record is removed.
- o Double-RRset: the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key, and the new DS record added to the parent zone. After waiting a suitable interval for the old DS and DNSKEY RRsets to expire from caches, the old DNSKEY and DS record are removed.

In essence, Double-KSK means that the new KSK is introduced first and used to sign the DNSKEY RRset. The DS record is changed, and finally the old KSK removed. It limits interactions with the parent to a minimum but, for the duration of the rollover, the size of the DNSKEY RRset is increased.

With Double-DS, the order of operations is the other way round:

introduce the new DS, change the DNSKEY, then remove the old DS. The size of the DNSKEY RRset is kept to a minimum, but two interactions are required with the parent.

Finally, Double-RRset is the fastest way to roll the KSK, but has the drawbacks of both of the other methods: a larger DNSKEY RRset and two interactions with the parent.

3. Key Rollover Timelines

3.1. Key States

DNSSEC validation requires both the DNSKEY and information created from it (referred to as "associated data" in this section). In the case of validation of an RR, the data associated with the the key is the corresponding RRSIG. Where there is a need to validate a chain of trust, the associated data is the DS record.

During the rolling process, keys move through different states. The defined states are:

Generated	Although keys may be created immediately prior to first use, some implementations may find it convenient to create a pool of keys in one operation and draw from it as required. (Note: such a pre-generated pool must be secured against surreptitious use.) Keys that have been created but not yet used are said to be in the "Generated" state.
Published	A key enters the published state when either it or its associated data first appears in the appropriate zone.
Ready	The DNSKEY or its associated data have been published for long enough to guarantee that copies of the key(s) it is replacing (or associated data related to that key) have expired from caches.
Active	The data is starting to be used for validation. In the case of a ZSK, it means that the key is now being be used to sign RRsets and that both it and the created RRSIGs appear in the zone. In the case of a KSK, it means that it is possible to use it to validate a DNSKEY RRset as both the DNSKEY and DS records are present in their relevant zones. Note that when this state is entered, it may not be possible for validating resolvers to use the data for validation in all cases: the zone signing may not have finished, or the data might not have reached the

resolver because of propagation delays and/or caching issues. If this is the case, the resolver will have to rely on the predecessor data instead.

Retired The data has ceased to be used for validation. In the case of a ZSK, it means that the key is no longer used to sign RRsets. In the case of a KSK, it means that the successor DNSKEY and DS records are in place. In both cases, the key (and its associated data) can be removed as soon as it is safe to do so, i.e. when all validating resolvers are able to use the new key and associated data to validate the zone. However, until this happens, the current key and associated data must remain in their respective zones.

Dead The key and its associated data are present in their respective zones, but there is no longer information anywhere that requires their presence for use in validation. Hence they can be removed at any time.

Removed Both the DNSKEY and its associated data have been removed from their respective zones.

There is one additional state, used where [RFC5011] considerations are in effect (see Section 3.3.4):

Revoked The DNSKEY is published for a period with the "revoke" bit set as a way of notifying validating resolvers that have configured it as an [RFC5011] trust anchor that it is about to be removed from the zone.

3.2. Zone-Signing Key Timelines

The following sections describe the rolling of a ZSK. They show the events in the lifetime of a key (referred to as "key N") and cover its replacement by its successor (key N+1).

3.2.1. Pre-Publication Method

In this method, the new key is introduced into the DNSKEY RRset. After enough time to ensure that any cached DNSKEY RRsets contain both keys, the zone is signed using the new key and the old signatures are removed. Finally, when all signatures created with the old key have expired from caches, the old key is removed.

The following diagram shows the timeline of a Pre-Publication rollover. Time increases along the horizontal scale from left to right and the vertical lines indicate events in the process.

Significant times and time intervals are marked.

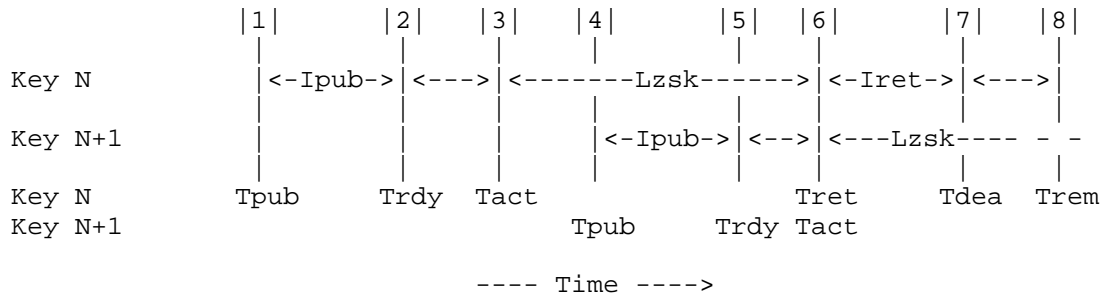


Figure 1: Timeline for a Pre-Publication ZSK rollover.

Event 1: Key N's DNSKEY record is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the currently active key-signing keys. The time at which this occurs is the publication time (Tpub), and the key is now said to be published. Note that the key is not yet used to sign records.

Event 2: Before it can be used, the key must be published for long enough to guarantee that any cached version of the zone's DNSKEY RRset includes this key.

This interval is the publication interval (Ipub) and, for the second or subsequent keys in the zone, is given by:

$$I_{pub} = D_{prp} + TTL_{key}$$

Here, Dprp is the propagation delay - the time taken for a change introduced at the master to replicate to all name servers. TTLkey is the time-to-live (TTL) for the DNSKEY records in the zone. The sum is therefore the maximum time taken for existing DNSKEY records to expire from caches, regardless of the nameserver from which they were retrieved.

(The case of introducing the first ZSK into the zone is discussed in Section 3.3.5.)

After a delay of Ipub, the key is said to be ready and could be used to sign records. The time at which this event occurs is key N's ready time (Trdy), which is given by:

$$\text{Trdy}(N) = \text{Tpub}(N) + \text{Ipub}$$

Event 3: At some later time, the key starts being used to sign RRs. This point is the activation time (Tact) and after this, key N is said to be active.

$$\text{Tact}(N) \geq \text{Trdy}(N)$$

Event 4: At some point a key must be given to its successor (key $N+1$). As with the introduction of the currently active key into the zone, the successor key will need to be published at least Ipub before it is activated. The publication time of key $N+1$ depends on the activation time of key N :

$$\text{Tpub}(N+1) \leq \text{Tact}(N) + \text{Lzsk} - \text{Ipub}$$

Here, Lzsk is the length of time for which a ZSK will be used (the ZSK lifetime). It should be noted that in the diagrams the actual key lifetime is represented; this may differ slightly from the intended lifetime set by key management policy.

Event 5: While key N is still active, its successor becomes ready. From this time onwards, key $N+1$ could be used to sign the zone.

Event 6: When key N has been in use for an interval equal to the ZSK lifetime, it is retired (i.e. it will never again be used to generate new signatures) and key $N+1$ activated and used to sign the zone. This is the retire time of key N (Tret), and is given by:

$$\text{Tret}(N) = \text{Tact}(N) + \text{Lzsk}$$

It is also the activation time of the successor key $N+1$. Note that operational considerations may cause key N to remain in use for a longer (or shorter) time than the lifetime set by the key management policy.

Event 7: The retired key needs to be retained in the zone whilst any RRSIG records created using this key are still published in the zone or held in caches. (It is possible that a validating resolver could have an old RRSIG record in the cache, but the old DNSKEY RRset has expired when it is asked to provide both to a client. In this case the DNSKEY RRset would need to be looked up again.) This means that once the key is no longer used to sign records, it should be retained in the zone for at least the retire interval (Iret) given by:

$$\text{Iret} = \text{Dsgn} + \text{Dprp} + \text{TTLsig}$$

Dsgn is the delay needed to ensure that all existing RRs have been

re-signed with the new key. Dprp is the propagation delay, required to guarantee that the updated zone information has reached all slave servers, and TTLsig is the maximum TTL of all the RRSIG records in the zone created with the retiring key.

The time at which all RRSIG records created with this key have expired from resolver caches is the dead time (Tdea), given by:

$$Tdea(N) = Tret(N) + Iret$$

... at which point the key is said to be dead.

Event 8: At any time after the key becomes dead, it can be removed from the zone's DNSKEY RRset, which must then be re-signed with the current key-signing key. This time is the removal time (Trem), given by:

$$Trem(N) \geq Tdea(N)$$

... at which time the key is said to be removed.

3.2.2. Double-Signature Method

In this rollover, a new key is introduced and used to sign the zone; the old key and signatures are retained. Once all cached DNSKEY and/or RRSIG information contains copies of the new DNSKEY and RRSIGs created with it, the old DNSKEY and RRSIGs can be removed from the zone.

The timeline for a double-signature rollover is shown below. The diagram follows the convention described in Section 3.2.1

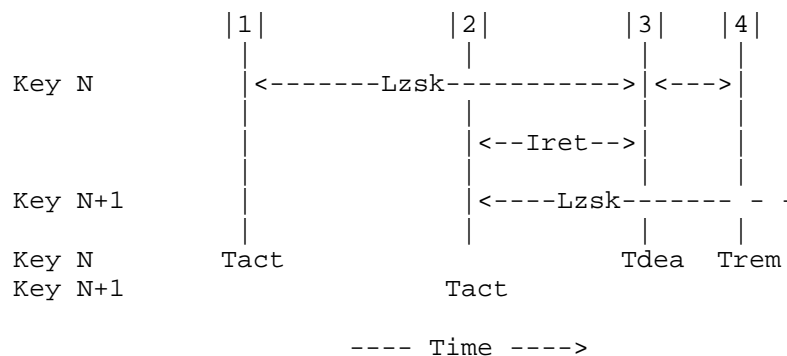


Figure 2: Timeline for a Double-Signature ZSK rollover.

Event 1: Key N is added to the DNSKEY RRset and is then used to sign the zone; existing signatures in the zone are not removed. The key is published and active: this is key N's activation time (Tact), after which the key is said to be active.

Event 2: As the current key (key N) approaches the end of its actual lifetime (Lzsk), the successor key (key N+1) is introduced into the zone and starts being used to sign RRsets: neither the current key nor the signatures created with it are removed. The successor key is now also active.

$$\text{Tact}(N+1) = \text{Tact}(N) + \text{Lzsk} - \text{Iret}$$

Event 3: Before key N can be withdrawn from the zone, all RRsets that need to be signed must have been signed by the successor key (key N+1) and any old RRsets that do not include the new key or new RRSIGs must have expired from caches. Note that the signatures are not replaced - each RRset is signed by both the old and new key.

This takes Iret, the retire interval, given by the expression:

$$\text{Iret} = \text{Dsgn} + \text{Dprp} + \max(\text{TTLkey}, \text{TTLsig})$$

As before, Dsgn is the delay needed to ensure that all existing RRsets have been signed with the new key and Dprp is the propagation delay, required to guarantee that the updated zone information has reached all slave servers. The final term (the maximum of TTLkey and TTLsig) is the period to wait for key and signature data associated with key N to expire from caches. (TTLkey is the TTL of the DNSKEY RRset and TTLsig is the maximum TTL of all the RRSIG records in the zone created with the ZSK. The two may be different: although the TTL of an RRSIG is equal to the TTL of the RRs in the associated RRset [RFC4034], the DNSKEY RRset only needs to be signed with the KSK.)

At the end of this interval, key N is said to be dead. This occurs at the dead time (Tdea) so:

$$\text{Tdea}(N) = \text{Tact}(N+1) + \text{Iret}$$

Event 4: At some later time key N and the signatures generated with it can be removed from the zone. This is the removal time (Trem), given by:

$$\text{Trem}(N) \geq \text{Tdea}(N)$$

3.3. Key-Signing Key Rollover Timelines

The following sections describe the rolling of a KSK. They show the events in the lifetime of a key (referred to as "key N") and cover its replacement by its successor (key N+1). (The case of introducing the first KSK into the zone is discussed in Section 3.3.5.)

3.3.1. Double-KSK Method

In this rollover, The new DNSKEY is added to the zone. After an interval long enough to guarantee that any cached DNSKEY RRsets contain the new DNSKEY, the DS record in the parent zone is changed. After a further interval to allow the old DS record to expire from caches, the old DNSKEY is removed from the zone.

The timeline for a Double-KSK rollover is shown below. The diagram follows the convention described in Section 3.2.1.

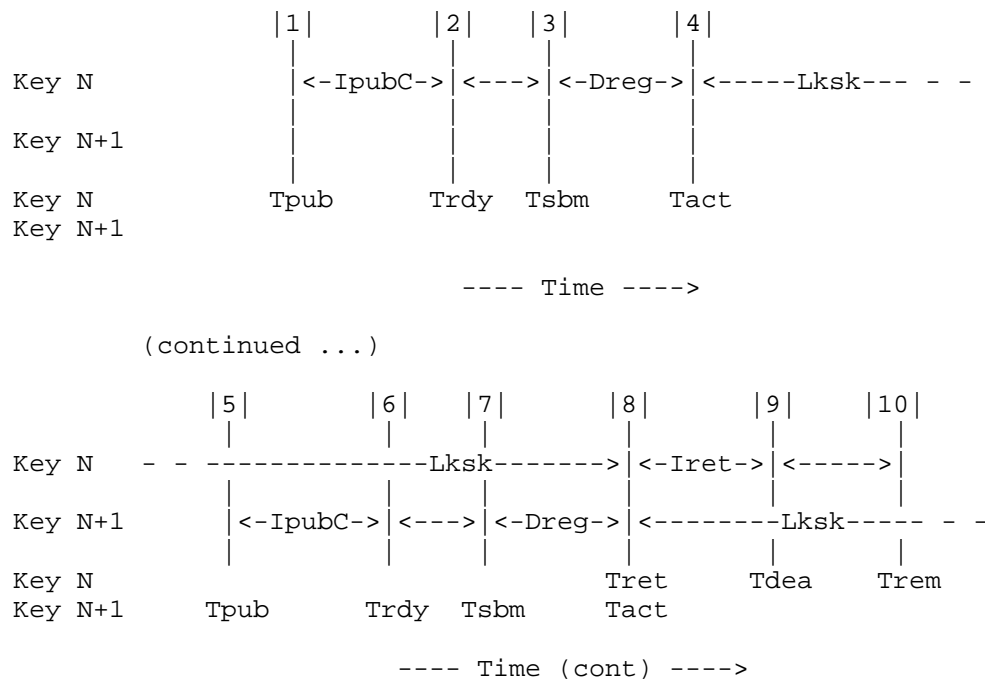


Figure 3: Timeline for a Double-KSK rollover.

Event 1: Key N is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by all currently active KSKs. (So at

this point, the DNSKEY RRset is signed by both key N and its predecessor KSK. If other KSKs were active, it is signed by these as well.) This is the publication time of key N (T_{pub}); after this the key is said to be published.

Event 2: Before it can be used, the key must be published for long enough to guarantee that any validating resolver that has a copy of the DNSKEY RRset in its cache will have a copy of the RRset that includes this key: in other words, that any prior cached information about the DNSKEY RRset has expired.

The interval is the publication interval in the child zone (I_{pubC}) and is given by:

$$I_{pubC} = D_{prpC} + TTL_{key}$$

... where D_{prpC} is the propagation delay for the child zone (the zone containing the KSK being rolled) and TTL_{key} the TTL for the DNSKEY RRset. The time at which this occurs is the key N's ready time, $Trdy$, given by:

$$Trdy(N) = T_{pub}(N) + I_{pubC}$$

Event 3: At some later time, the DS record corresponding to the new KSK is submitted to the parent zone for publication. This time is the submission time, T_{sbm} :

$$T_{sbm}(N) \geq Trdy(N)$$

Event 4: The DS record is published in the parent zone. As this is the point at which all information for authentication - both DNSKEY and DS record - is available in the two zones, in analogy with other rollover methods, this is called the activation time of key N (T_{act}):

$$T_{act}(N) = T_{sbm}(N) + D_{reg}$$

... where D_{reg} is the registration delay, the time taken after the DS record has been submitted to the parent zone manager for it to be placed in the zone. (Parent zones are often managed by different entities, and this term accounts for the organisational overhead of transferring a record. In practice, D_{reg} will not be a fixed time: instead, the end of D_{reg} will be signalled by the appearance of the DS record in the parent zone.)

Event 5: While key N is active, thought needs to be given to its successor (key N+1). At some time before the scheduled end of the KSK lifetime, the successor KSK is published in the zone. (As before, this means that the DNSKEY RRset is signed by all KSKs.)

This time is the publication time of the successor key N+1, given by:

$$T_{\text{pub}}(N+1) \leq T_{\text{act}}(N) + L_{\text{sk}} - D_{\text{reg}} - I_{\text{pubC}}$$

... where L_{sk} is the actual lifetime of the KSK, and D_{reg} the registration delay.

Event 6: After an interval I_{pubC} , key N+1 becomes ready (in that all caches that have a copy of the DNSKEY RRset have a copy of this key). This time is the ready time of the successor key N+1 (T_{rdy}).

Event 7: At the submission time of the successor key N+1, $T_{\text{sbm}}(N+1)$, the DS record corresponding to key N+1 is submitted to the parent zone.

Event 8: The successor DS record is published in the parent zone and the current DS record withdrawn. Key N is said to be retired and the time at which this occurs is $T_{\text{ret}}(N)$, given by:

$$T_{\text{ret}}(N) = T_{\text{sbm}}(N+1) + D_{\text{reg}}$$

Event 9: Key N must remain in the zone until any caches that contain a copy of the DS RRset have a copy containing the new DS record. This interval is the retire interval, given by:

$$I_{\text{ret}} = D_{\text{prpP}} + T_{\text{TLDs}}$$

... where D_{prpP} is the propagation delay in the parent zone and T_{TLDs} the TTL of a DS record in the parent zone.

As the key is no longer used for anything, it is said to be dead. This point is the dead time (T_{dea}), given by:

$$T_{\text{dea}}(N) = T_{\text{ret}}(N) + I_{\text{ret}}$$

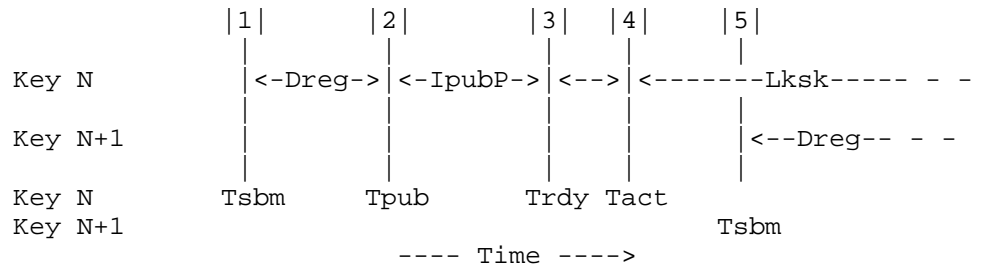
Event 10: At some later time, key N is removed from the zone's DNSKEY RRset (at the remove time T_{rem}); the key is now said to be removed.

$$T_{\text{rem}}(N) \geq T_{\text{dea}}(N)$$

3.3.2. Double-DS Method

In this rollover, the new DS record is published in the parent zone. When any caches that contain the DS RRset contain a copy of the new record, the KSK in the zone is changed. After a further interval for the old DNSKEY RRset to expire from caches, the old DS record is removed from the parent.

The timeline for a Double-DS rollover is shown below. The diagram follows the convention described in Section 3.2.1



(continued ...)

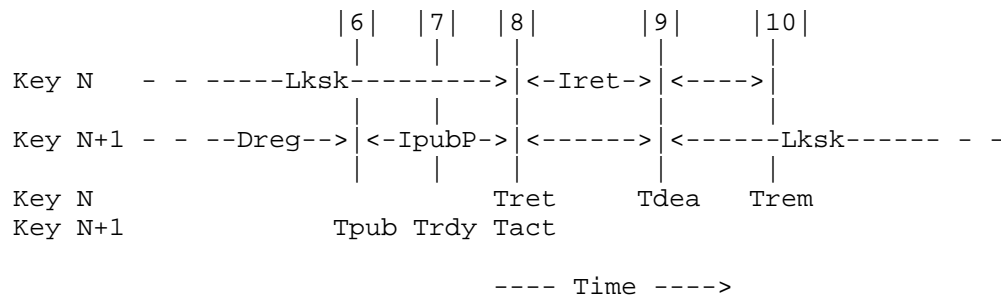


Figure 4: Timeline for a Double-DS KSK rollover.

Event 1: The DS RR is submitted to the parent zone for publication. This time is the submission time, T_{sbm} .

Event 2: After the registration delay, Dreg, the DS record is published in the parent zone. This is the publication time (T_{pub}) of key N, given by:

$$T_{pub}(N) = T_{sbm}(N) + Dreg$$

As before, in practice Dreg will not be a fixed time. Instead, the end of Dreg will be signalled by the appearance of the DS record in the parent zone.

Event 3: At some later time, any cache that has a copy of the DS RRset will have a copy of the DS record for key N. At this point, key N, if introduced into the DNSKEY RRset, could be used to validate the zone. For this reason, this time is known as the ready time, T_{rdy} , and is given by:

$$\text{Trdy}(N) = \text{Tpub}(N) + \text{IpubP}$$

IpubP is the publication interval of the DS record (in the parent zone) and is given by the expression:

$$\text{IpubP} = \text{DprpP} + \text{TTLds}$$

... where DprpP is the propagation delay for the parent zone and TTLds the TTL assigned to DS records in that zone.

Event 4: At some later time, the key rollover takes place and the new key (key N) is introduced into the DNSKEY RRset and used to sign it. This time is key N's activation time (Tact) and at this point key N is said to be active:

$$\text{Tact}(N) \geq \text{Trdy}(N)$$

Event 5: At some point thought must be given to key replacement. The DS record for the successor key must be submitted to the parent zone at a time such that when the current key is withdrawn, any cache that contains the zone's DS records has data about the DS record of the successor key. The time at which this occurs is the submission time of the successor key N+1, given by:

$$\text{Tsbm}(N+1) \leq \text{Tact}(N) + \text{Lksk} - \text{IpubP} - \text{Dreg}$$

... where Lksk is the actual lifetime of key N (which may differ slightly from the lifetime set in the key management policy) and Dreg is the registration delay.

Event 6. After an interval Dreg, the successor DS record is published in the zone.

Event 7: The successor key (key N+1) enters the ready state, i.e. its DS record is now in caches that contain the parent DS RRset.

Event 8: When key N has been active for its lifetime (Lksk), it is replaced in the DNSKEY RRset by key N+1; the RRset is then signed with the new key. At this point, as both the old and new DS records have been in the parent zone long enough to ensure that they are in caches that contain the DS RRset, the zone can be authenticated throughout the rollover. A validating resolver can authenticate either the old or new KSK.

This time is the retire time (Tret) of key N, given by:

$$Tret(N) = Tact(N) + Lksk$$

This is also the activation time of the successor key N+1.

Event 9: At some later time, all copies of the old DNSKEY RRset have expired from caches and the old DS record is no longer needed. In analogy with other rollover methods, this is called the dead time, Tdea, and is given by:

$$Tdea(N) = Tret(N) + Iret$$

... where Iret is the retire interval of the key, given by:

$$Iret = DprpC + TTLkey$$

As before, this term includes DprpC, the time taken to propagate the RRset change through the master-slave hierarchy of the child zone and TTLkey, the time taken for the DNSKEY RRset to expire from caches.

Event 10: At some later time, the DS record is removed from the parent zone. In analogy with other rollover methods, this is the removal time (Trem), given by:

$$Trem(N) \geq Tdea(N)$$

3.3.3. Double-RRset Method

In the Double-RRset rollover, the new DNSKEY and DS records are published simultaneously in the appropriate zones. Once enough time has elapsed for the old DNSKEY and DS RRsets to expire from caches, the old DNSKEY and DS records are removed from their respective zones.

The timeline for this rollover is shown below. The diagram follows the convention described in Section 3.2.1

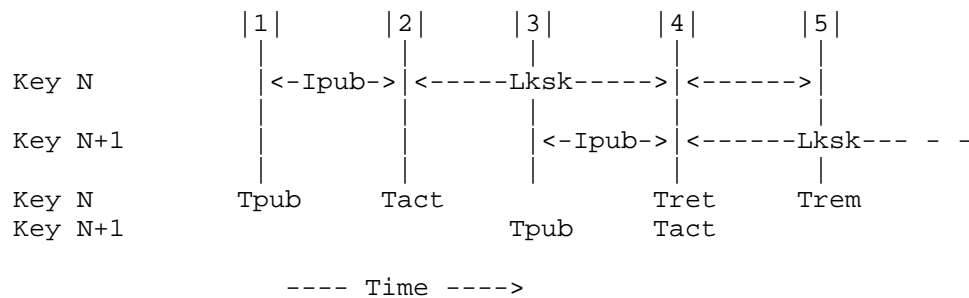


Figure 5: Timeline for a Double-RRset KSK rollover.

Event 1: The key is added to and used for signing the DNSKEY RRset and is thereby published in the zone. At the same time the corresponding DS record is submitted to the parent zone for publication. This time is the publish time for key N (T_{pub}) and the key is said to be published.

Event 2: At some later time, the DS record is published in the parent zone and at some time after that, the updated information has reached all caches: any cache that holds a DNSKEY RRset from the child zone will have a copy that includes the new KSK, and any cache that has a copy of the parent DS RRset will have a copy that includes the new DS record.

The time at which this occurs is called the activation time of key N (T_{act}), given by:

$$T_{act}(N) = T_{pub}(N) + I_{pub}$$

... where I_{pub} is the composite publication interval for the DNSKEY and DS records, given by:

$$I_{pub} = \max(I_{pubP}, I_{pubC}),$$

I_{pubP} being the publication interval of the DS record in the parent zone and I_{pubC} the publication interval of the DNSKEY in the child zone. The parent zone's publication interval is given by:

$$I_{pubP} = D_{reg} + D_{prpP} + TTL_{ds}$$

where D_{reg} is the registration delay, the time taken for the DS record to be published in the parent zone. D_{prpP} is the parent zone's propagation delay and TTL_{ds} is the TTL of the DS record in that zone.

The child zone's publication interval is given by a similar equation:

$$\text{IpubC} = \text{DprpC} + \text{TTLkey}$$

... where DprpC is the propagation delay in the child zone and TTLkey the TTL of a DNSKEY record.

Event 3: At some point we need to give thought to key replacement. The successor key (key N+1) must be introduced into the zone (and its DS record submitted to the parent) at a time such that it becomes active when the current key has been active for its actual lifetime, Lksk. This is the publication time (Tpub) of the successor key, and is given by:

$$\text{Tpub}(N+1) \leq \text{Tact}(N) + \text{Lksk} - \text{Ipub}$$

... where Lksk is the actual lifetime of the KSK and Ipub is as defined above.

Event 4: Key N+1's DNSKEY and DS records are now in caches that contain the child zone DNSKEY and/or the parent zone DS RR, and so the zone can be validated with the new key. This is the activation time (Tact) of the successor key N+1 and by analogy with other rollover methods, it is also the dead time of key N:

$$\text{Tdea}(N) = \text{Tact}(N) + \text{Lksk}$$

Event 5: At some later time, the key N's DS and DNSKEY records are removed from their respective zones. In analogy with other rollover methods, this is the removal time (Trem), given by:

$$\text{Trem}(N) \geq \text{Tdea}(N)$$

3.3.4. Interaction with Configured Trust Anchors

Although the preceding sections have been concerned with rolling KSKs where the trust anchor is a DS record in the parent zone, zone managers may want to take account of the possibility that some validating resolvers may have configured trust anchors directly.

Rolling a configured trust anchor is dealt with in [RFC5011]. It requires introducing the KSK to be used as the trust anchor into the zone for a period of time before use, and retaining it (with the "revoke" bit set) for some time after use.

3.3.4.1. Addition of KSK

When the new key is introduced, the expression for the publication interval of the DNSKEY(IpubC) in the Double-KSK and Double-RRset methods is modified to:

$$\text{IpubC} \geq \text{DprpC} + \max(\text{Itrp}, \text{TTLkey})$$

... where the right hand side of the expression now includes the "trust point" interval. This term is the interval required to guarantee that a resolver configured for the automatic update of keys from a particular trust point will see at least two validated DNSKEY RRsets containing the new key (a requirement from [RFC5011], section 2.4.1). It is defined by the expression:

$$\text{Itrp} \geq (2 * \text{queryInterval}) + (n * \text{retryTime})$$

... where queryInterval and retryTime are as defined in section 2.3 of [RFC5011]. "n" is the total number of retries needed by the resolver during the two attempts to get the DNSKEY RRset.

The first term of the expression $(2 * \text{queryInterval})$ represents the time to obtain two validated DNSKEY RRsets. The second term $(n * \text{retryTime})$ is a safety margin, with the value of "n" reflecting the degree of confidence in the communication between a resolver and the trust point.

In the Double-DS method, instead of swapping the KSK RRs in a single step, there must now be a period of overlap. In other words, the new KSK must be introduced into the zone at least:

$$\text{DprpC} + \max(\text{Itrp}, \text{TTLkey})$$

... before the switch is made.

3.3.4.2. Removal of KSK

The timeline for the removal of the key in all methods is modified by introducing a new state, "revoked". When the key reaches its dead time, instead of being declared "dead", it is revoked; the "revoke" bit is set in the published DNSKEY RR, and the DNSKEY RRset re-signed with the current and revoked keys. The key is maintained in this state for the "revoke" interval, Irev, given by:

$$\text{Irev} \geq 30 \text{ days}$$

... 30 days being the [RFC5011] remove hold-down time. After this time, the key is dead and can be removed from the zone.

3.3.5. Introduction of First Keys

There are no timing considerations associated with the introduction of the first keys into a zone other than that they must be introduced and the zone validly signed before a chain of trust to the zone is

created.

This is important: in the case of a secure parent, it means ensuring that the DS record is not published in the parent zone until there is no possibility that a validating resolver can obtain the record yet is not able to obtain the corresponding DNSKEY. In the case of an insecure parent, i.e. the initial creation of a new security apex, it is not possible to guarantee this. It is up to the operator of the validating resolver to wait for the new KSK to appear at all servers for the zone before configuring the trust anchor.

4. Standby Keys

Although keys will usually be rolled according to some regular schedule, there may be occasions when an emergency rollover is required, e.g., if the active key is suspected of being compromised. The aim of the emergency rollover is to allow the zone to be re-signed with a new key as soon as possible. As a key must be in the ready state to sign the zone, having at least one additional key (a standby key) in this state at all times will minimise delay.

In the case of a ZSK, a standby key only really makes sense with the Pre-Publication method. A permanent standby DNSKEY RR should be included in the zone or successor keys could be introduced as soon as possible after a key becomes active. Either way results in one or more additional ZSKs in the DNSKEY RRset that can immediately be used to sign the zone if the current key is compromised.

(Although in theory the mechanism could be used with both the Double-Signature and Double-RRSIG methods, it would require pre-publication of the signatures. Essentially, the standby key would be permanently active, as it would have to be periodically used to renew signatures. Zones would also permanently require two sets of signatures.)

It is also possible to have a standby KSK. The Double-KSK method requires that the standby KSK be included in the DNSKEY RRset; rolling the key then requires just the introduction of the DS record in the parent. Note that the standby KSK should also be used to sign the DNSKEY RRset. As the RRset and its signatures travel together, merely adding the KSK without using it to sign the DNSKEY RRset does not provide the desired time saving: for a KSK to be used in a rollover the DNSKEY RRset must be signed with it, and this would introduce a delay while the old RRset (not signed with the new key) expires from caches.

The idea of a standby KSK in the Double-RRset rollover method effectively means having two active keys (as the standby KSK and

associated DS record would both be published at the same time in their respective zones).

Finally, in the Double-DS method of rolling a KSK, it is not a standby key that is present, it is a standby DS record in the parent zone.

Whatever algorithm is used, the standby item of data can be included in the zone on a permanent basis, or be a successor introduced as early as possible.

5. Algorithm Considerations

The preceding sections have implicitly assumed that all keys and signatures are created using a single algorithm. However, [RFC4035] (section 2.2) requires that there be an RRSIG for each RRset using at least one DNSKEY of each algorithm in the zone apex DNSKEY RRset.

Except in the case of an algorithm rollover - where the algorithms used to create the signatures are being changed - there is no relationship between the keys of different algorithms. This means that they can be rolled independently of one another. In other words, the key rollover logic described above should be run separately for each algorithm; the union of the results is included in the zone, which is signed using the active key for each algorithm.

6. Summary

For ZSKs, "Pre-Publication" is generally considered to be the preferred way of rolling keys. As shown in this document, the time taken to roll is wholly dependent on parameters under the control of the zone manager.

In contrast, "Double-RRset" is the most efficient method for KSK rollover due to the ability to have new DS records and DNSKEY RRsets propagate in parallel. The time taken to roll KSKs may depend on factors related to the parent zone if the parent is signed. For zones that intend to comply with the recommendations of [RFC5011], in virtually all cases the rollover time will be determined by the RFC5011 "add hold-down" and "remove hold-down" times. It should be emphasized that this delay is a policy choice and not a function of timing values and that it also requires changes to the rollover process due to the need to manage revocation of trust anchors.

Finally, the treatment of emergency key rollover is significantly simplified by the introduction of standby keys as standard practice

during all types of rollovers.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document does not introduce any new security issues beyond those already discussed in [RFC4033], [RFC4034], [RFC4035] and [RFC5011].

9. Acknowledgements

The authors gratefully acknowledge help and contributions from Roy Arends and Wouter Wijngaards.

10. Normative References

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, September 2007.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, December 2012.

Appendix A. List of Symbols

The document defines a number of symbols, all of which are listed here. All are of the form:

All symbols used in the text are of the form:

<TYPE><id><ZONE>

where:

<TYPE> is an upper-case character indicating what type the symbol is. Defined types are:

D	delay: interval that is a feature of the process
I	interval between two events
L	lifetime: interval set by the zone manager
T	a point in time
TTL	TTL of a record

I, T and TTL are self-explanatory. Like I, both D and L are time periods, but whereas I values are intervals between two events (even if the events are defined in terms of the interval, e.g., the dead time occurs "retire interval" after the retire time), D and L are fixed intervals: a "D" interval (delay) is a feature of the process, probably outside control of the zone manager, whereas an "L" interval (lifetime) is chosen by the zone manager and is a feature of policy.

<id> is lower-case and defines what object or event the variable is related to, e.g.,

act	activation
pub	publication
ret	retire

<ZONE> is an optional capital letter that distinguishes between the same variable applied to different zones and is one of:

C	child
P	parent

Within the rollover descriptions, times may be suffixed by a number in brackets indicating the instance of the key to which they apply, e.g. Tact(N) is the activation time of key N, Tpub(N+1) the publication time of key N+1 etc.

The list of variables used in the text given below.

Dprp	Propagation delay. The amount of time for a change made at a master nameserver to propagate to all the slave nameservers.
DprpC	Propagation delay in the child zone.
DprpP	Propagation delay in the parent zone.
Dreg	Registration delay: the time taken for a DS record submitted to a parent zone to appear in it. As a parent zone is often managed by a different organisation to that managing the child zone, the delays associated with passing data between organisations is captured by this term.
Dsgn	Signing delay. After the introduction of a new ZSK, the amount of time taken for all the RRs in the zone to be signed with it.
Ipub	Publication interval. The amount of time that must elapse after the publication of a DNSKEY and/or its associated data before it can be assumed that any resolvers that have the relevant RRset cached have a copy of the new information.
IpubC	Publication interval in the child zone.
IpubP	Publication interval in the parent zone.
Iret	Retire interval. The amount of time that must elapse after a DNSKEY or associated data enters the retire state for any dependent information (e.g. RRSIG for a ZSK) to be purged from validating resolver caches.
Irev	Revoke interval. The amount of time that a KSK must remain published with the revoke bit set to satisfy [RFC5011] considerations.
Itrp	Trust-point interval. The amount of time that a trust anchor must be published for to guarantee that a resolver configured for an automatic update of keys will see the new key at least twice.
Lksk	Lifetime of a key-signing key. This is the actual amount of time for which this particular KSK is regarded as the active KSK. Depending on when the key is rolled-over, the actual lifetime may be longer or shorter than the intended key lifetime indicated by management policy.

Lzsk	Lifetime of a zone-signing key. This is the actual amount of time for which the ZSK is used to sign the zone. Depending on when the key is rolled-over, the actual lifetime may be longer or shorter than the intended key lifetime indicated by management policy.
Tact	Activation time. The time at which the key is regarded as the principal key for the zone.
Tdea	Dead time. The time at which any information held in validating resolver caches is guaranteed to contain information related to the successor key. At this point, the current key and its associated information are not longed required for validation purposes.
Tpub	Publication time. The time that the key or associated data appears in the zone for the first time.
Trem	Removal time. The time at which the key and its associated information starts being removed from their respective zones.
Tret	Retire time. The time at which successor information starts being used.
Trdy	Ready time. The time at which it can be guaranteed that validating resolvers that have information about the key and/or associated data cached have a copy of the new information.
Tsbm	Submission time. The time at which the DS record of a KSK is submitted to the parent zone.
TTLds	Time to live of a DS record.
TTLkey	Time to live of a DNSKEY record. (By implication, this is also the time to live of the signatures on the DNSKEY RRset.)
TTLsig	The maximum time to live of all the RRSIG records in the zone that were created with the ZSK.

Appendix B. Change History (To be removed on publication)

- o draft-ietf-dnsop-dnssec-key-timing-06
 - * Clarifications to various text, as identified in WGLC.
 - * Moved "Limitation of Scope" section to be a subsection of

section 1.

- o draft-ietf-dnsop-dnssec-key-timing-05
 - * Some more renamings of "Double-Signature" KSK rollover to "Double-KSK".
 - * Remove Tgen from diagrams.
 - * Review by Richard Lamb.
 - * Updated KSK rollover summary text.
 - * Updated variable descriptions in the appendix.
- o draft-ietf-dnsop-dnssec-key-timing-04
 - * Renamed to "DNSSEC Key Rollover Timing Considerations" to emphasise that this draft concerns rollover timings.
 - * Updated 4641bis reference to RFC 6781.
 - * Add introductory paragraph to each rollover description summarising its essential features.
 - * Remove detailed description of double-RRSIG ZSK rollover. It is extremely unlikely to be used in any practical situation.
 - * "Double-Signature" KSK rollover renamed to "Double-KSK" to avoid confusion with the ZSK rollover of the same name.
 - * Removed section 2.3 (rollover summary) which just listed the order in which records are published.
 - * Matthijs Mekking added as co-author.
 - * Changed Lzsk and Kzsk definitions: actual lifetime instead of intended lifetime.
 - * Update diagrams and text to better reflect key states and key lifetimes.
- o draft-ietf-dnsop-dnssec-key-timing-03
 - * Clarifications of and corrections to wording (Marc Lampo, Alfred Hoenes).
 - * Updated timings related to trust anchor interaction (Matthijs Mekking).
 - * Updated RFC 4641 reference to 4641bis (Alfred Hoenes).
 - * Moved change history to end of document (Alfred Hoenes).
- o draft-ietf-dnsop-dnssec-key-timing-02
 - * Significant re-wording of some sections.
 - * Removal of events noting change of state of predecessor key from ZSK Double-RRSIG and Double-Signature methods.
 - * Change order of bullet points (and some wording) in section 1.1.
 - * Remove discussion of advantages and disadvantages of key roll methods from section 2: draft is informative and does not give recommendations.
 - * Removal of discussion of upper limit to retire time relationship to signature lifetime.
 - * Remove timing details of first key in the zone and move discussion of first signing of a zone to later in the document.

(Matthijs Mekking)

* Removal of redundant symbols from Appendix A.

- o draft-ietf-dnsop-dnssec-key-timing-01
 - * Added section on limitation of scope.
- o draft-ietf-dnsop-dnssec-key-timing-00
 - * Change to author contact details.
- o draft-morris-dnsop-dnssec-key-timing-02
 - * General restructuring.
 - * Added descriptions of more rollovers (IETF-76 meeting).
 - * Improved description of key states and removed diagram.
 - * Provided simpler description of standby keys.
 - * Added section concerning first key in a zone.
 - * Moved [RFC5011] to a separate section.
 - * Various nits fixed (Alfred Hoenes, Jeremy Reed, Scott Rose, Sion Lloyd, Tony Finch).
- o draft-morris-dnsop-dnssec-key-timing-01
 - * Use latest boilerplate for IPR text.
 - * List different ways to roll a KSK (acknowledgements to Mark Andrews).
 - * Restructure to concentrate on key timing, not management procedures.
 - * Change symbol notation (Diane Davidowicz and others).
 - * Added key state transition diagram (Diane Davidowicz).
 - * Corrected spelling, formatting, grammatical and style errors (Diane Davidowicz, Alfred Hoenes and Jinmei Tatuya).
 - * Added note that in the case of multiple algorithms, the signatures and rollovers for each algorithm can be considered as more or less independent (Alfred Hoenes).
 - * Take account of the fact that signing a zone is not atomic (Chris Thompson).
 - * Add section contrasting pre-publication rollover with double signature rollover (Matthijs Mekking).
 - * Retained distinction between first and subsequent keys in definition of initial publication interval (Matthijs Mekking).
- o draft-morris-dnsop-dnssec-key-timing-00
 - Initial draft.

Authors' Addresses

Stephen Morris
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
USA

Email: stephen@isc.org
URI: <http://www.isc.org>

Johan Ihren
Netnod
Franzengatan 5
Stockholm, SE-112 51
Sweden

Email: johani@netnod.se
URI: <http://www.netnod.se>

John Dickinson
Sinodun Internet Technologies Ltd
Magdalen Centre
Oxford Science Park
Robert Robertson Avenue
Oxford, Oxfordshire OX4 4GA
UK

Email: jad@sinodun.com
URI: <http://www.sinodun.com>

W. (Matthijs) Mekking
NLnet Labs
Science Park 400
Amsterdam 1098 XH
The Netherlands

Email: matthijs@nlnetlabs.nl
URI: <http://www.nlnetlabs.nl>

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: June 27, 2017

P. Koch
DENIC eG
M. Larson
P. Hoffman
ICANN
December 24, 2016

Initializing a DNS Resolver with Priming Queries
draft-ietf-dnsop-resolver-priming-11

Abstract

This document describes the queries that a DNS resolver should emit to initialize its cache. The result is that the resolver gets both a current NS RRSet for the root zone and the necessary address information for reaching the root servers.

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 June 27, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

Recursive DNS resolvers need a starting point to resolve queries. [RFC1034] describes a common scenario for recursive resolvers: they begin with an empty cache and some configuration for finding the names and addresses of the DNS root servers. [RFC1034] describes that configuration as a list of servers that will give authoritative answers to queries about the root. This has become a common implementation choice for recursive resolvers, and is the topic of this document.

This document describes the steps needed for this common implementation choice. Note that this is not the only way to start a recursive name server with an empty cache, but it is the only one described in [RFC1034]. Some implementers have chosen other directions, some of which work well and others of which fail (sometimes disastrously) under different conditions. For example, an implementation that only gets the addresses of the root name servers from configuration, not from the DNS as described in this document, will have stale data that could cause slower resolution.

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 [RFC2119].

This document only deals with recursive name servers (recursive resolvers, resolvers) for the IN class.

2. Description of Priming

Priming is the act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers. A recursive resolver starts with no information about the root servers, and ends up with a list of their names and their addresses.

Priming is described in Sections 5.3.2 and 5.3.3 of [RFC1034]. The scenario used in that description, that of a recursive server that is also authoritative, is no longer as common.

The configured list of IP addresses for the root servers usually comes from the vendor or distributor of the recursive server software. This list is usually correct and complete when shipped, but may become out of date over time.

The list of root server operators and the domain name associated with each one has been stable since 1997. However, there are address changes for the root server domain names, both for IPv4 and IPv6 addresses. However, research shows that after those addresses change, some resolvers never get the new addresses. Therefore, it is important that resolvers be able to cope with change, even without relying upon configuration updates to be applied by their operator. Root server change is the main reason that resolvers need to do priming instead of just going from a configured list to get a full and accurate list of root servers.

3. Priming Queries

A priming query is a DNS query used to get the root server information in a resolver. It has a QNAME of "." and a QTYPE of NS, and is sent to one of the addresses in the configuration for the recursive resolver. The priming query can be sent over either UDP or TCP. If the query is sent over UDP, the source port SHOULD be randomly selected (see [RFC5452]). The RD bit MAY be set to 0 or 1, although the meaning of it being set to 1 is undefined for priming queries.

The recursive resolver SHOULD use EDNS0 [RFC6891] for priming queries and SHOULD announce and handle a reassembly size of at least 1024 octets [RFC3226]. Doing so allows responses that cover the size of a full priming response (see Section 4.2) for the current set of root servers. See Section 3.3 for discussion of setting the DNSSEC OK (DO) bit (defined in [RFC4033]).

3.1. Repeating Priming Queries

The recursive resolver SHOULD send a priming query only when it is needed, such as when the resolver starts with an empty cache and when the NS RRset for the root zone has expired. Because the NS records for the root are not special, the recursive resolver expires those NS records according to their TTL values. (Note that a recursive resolver MAY pre-fetch the NS RRset before it expires.)

If a priming query does not get a response, the recursive resolver needs to retry the query with a different target address from the configuration.

3.2. Target Selection

In order to spread the load across all the root server domain names, the recursive resolver SHOULD select the target for a priming query randomly from the list of addresses. The recursive resolver might choose either IPv4 and IPv6 addresses based on its knowledge of

whether the system on which it is running has adequate connectivity on either type of address.

Note that this recommended method is not the only way to choose from the list in a recursive resolver's configuration. Two other common methods include picking the first from the list, and remembering which address in the list gave the fastest response earlier and using that one. There are probably other methods in use today. However, the random method listed above SHOULD be used for priming.

3.3. DNSSEC with Priming Queries

The resolver MAY set the DNSSEC OK (DO) bit. At the time this document is being published, there is little use to performing DNSSEC validation on the priming query. Currently all root name server names end in "root-servers.net" and the AAAA and A RRsets for the root server names reside in the "root-servers.net" zone. All root servers are also authoritative for this zone, allowing priming responses to include the appropriate root name server A and AAAA RRsets. But because the "root-servers.net" zone is not currently signed, these RRsets cannot be validated.

A man-in-the-middle attack on the priming query could direct a resolver to a rogue root name server. Note, however, that a validating resolver will not accept responses from rogue root name servers if they are different from the real responses because the resolver has a trust anchor for the root and the answers from the root are signed. Thus, if there is a man-in-the-middle attack on the priming query, the only result for a validating resolver will be a denial of service, not the resolver's accepting the bad responses.

If the "root-servers.net" zone is later signed, or if the root servers are named in a different zone and that zone is signed, having DNSSEC validation for the priming queries might be valuable.

4. Priming Responses

A priming query is a normal DNS query. Thus, a root name server cannot distinguish a priming query from any other query for the root NS RRSet. Thus, the root server's response will also be a normal DNS response.

4.1. Expected Properties of the Priming Response

The priming response is expected to have an RCODE of NOERROR, and to have the AA bit set. Also, it is expected to have an NS RRSet in the Answer section (because the NS RRSet originates from the root zone), and an empty Authority section (because the NS RRSet already appears

in the Answer section). There will also be an Additional section with A and/or AAAA RRSets for the root name servers pointed at by the NS RRSets.

Resolver software SHOULD treat the response to the priming query as a normal DNS response, just as it would use any other data fed to its cache. Resolver software SHOULD NOT expect exactly 13 NS RRs because historically some root servers have returned fewer.

4.2. Completeness of the Response

There are currently 13 root servers. All have one IPv4 address and one IPv6 address. Not even counting the NS RRSets, the combined size of all the A and AAAA RRSets exceeds the original 512 octet payload limit from [RFC1035].

In the event of a response where the Additional section omits certain root server address information, re-issuing of the priming query does not help with those root name servers that respond with a fixed order of addresses in the Additional section. Instead, the recursive resolver needs to issue direct queries for A and AAAA RRSets for the remaining names. Currently, these RRSets would be authoritatively available from the root name servers.

5. Security Considerations

Spoofing a response to a priming query can be used to redirect all of the queries originating from a victim recursive resolver to one or more servers for the attacker. Until the responses to priming queries are protected with DNSSEC, there is no definitive way to prevent such redirection.

An on-path attacker who sees a priming query coming from a resolver can inject false answers before a root server can give correct answers. If the attacker's answers are accepted, this can set up the ability to give further false answers for future queries to the resolver. False answers for root servers are more dangerous than, say, false answers for TLDs, because the root is the highest node of the DNS. See Section 3.3 for more discussion.

In both of the scenarios above, a validating resolver will be able to detect the attack if its chain of queries comes to a zone that is signed, but not for those that are unsigned.

6. IANA Considerations

None.

7. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver message size requirements", RFC 3226, DOI 10.17487/RFC3226, December 2001, <<http://www.rfc-editor.org/info/rfc3226>>.
- [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, <<http://www.rfc-editor.org/info/rfc4033>>.
- [RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <<http://www.rfc-editor.org/info/rfc5452>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<http://www.rfc-editor.org/info/rfc6891>>.

Appendix A. Acknowledgements

This document is the product of the DNSOP WG and benefitted from the reviews done there.

Authors' Addresses

Peter Koch
DENIC eG
Kaiserstrasse 75-77
Frankfurt 60329
DE

Phone: +49 69 27235 0
Email: pk@DENIC.DE

Matt Larson
ICANN

Email: matt.larson@icann.org

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 17, 2014

P. Vixie
Farsight Security, Inc.
A. Kato
Keio University/WIDE Project
J. Abley
Dyn, Inc.
February 13, 2014

DNS Referral Response Size Issues
draft-ietf-dnsop-respsize-15

Abstract

With a mandated default minimum maximum UDP message size of 512 octets, the DNS protocol presents some special problems for zones wishing to expose a moderate or high number of authority servers (NS resource records). This document explains the operational issues caused by, or related to this response size limit, and suggests ways to optimize the use of this limited space. Guidance is offered to DNS server implementors and to DNS zone administrators.

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 August 17, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Introduction and Overview	4
3. Delegation Details	5
3.1. Relevant Protocol Elements	5
3.2. Advice to Zone Administrators	6
3.3. Advice to Server Implementors	7
4. Analysis	9
5. Conclusions	12
6. Security Considerations	13
7. IANA Considerations	14
8. Acknowledgements	15
9. References	16
9.1. Normative References	16
9.2. Informative References	16
Appendix A. The response simulator program	18
Appendix B. Editorial Notes	20
B.1. Change History	20
Authors' Addresses	21

1. Terminology

This document uses terminology specific to the Domain Name System (DNS), including the following common abbreviations:

A: A resource record type used to specify an IPv4 address [RFC1034]

AAAA: A resource record type used to specify an IPv6 address [RFC3596]

CNAME: A resource record type used to define a canonical name [RFC1034]

DNAME: A resource record type used to map a DNS subtree onto another domain [RFC2672]

DNSSEC: DNS Security Extensions [RFC4033]

DO: "DNS OK" -- a flag in the EDNS header used to signal the ability to use DNSSEC [RFC4035]

EDNS: Extension mechanisms for DNS [RFC6891]

EDNS0: EDNS version 0 [RFC6891]

MTU: Maximum Transmission Unit, the maximum size for a datagram to be forwarded on an interface without needing fragmentation [RFC0791] [RFC2460]

NS: A resource record type used to specify a nameserver on either side of a zone cut [RFC1034]

RR: Resource Record [RFC1034]

RRSet: Resource Record Set [RFC1034]

TC: A bit in the DNS message header used to indicate that the message has been truncated [RFC1034]

In an exchange of DNS messages between two hosts, this document refers to the host sending a DNS request as the initiator, and the host sending a DNS response as the responder.

2. Introduction and Overview

The original DNS standard limited the UDP message size to 512 octets (see Section 4.2.1 of [RFC1035]). Even though this limitation was due to the required minimum IP reassembly limit for IPv4, it became a hard DNS protocol limit and is not implicitly relaxed by changes in a network layer protocol, e.g. by the larger minimum MTU specified in IPv6 [RFC2460] than in IPv4 [RFC0791].

The EDNS protocol extension starting with version 0 permits larger responses by mutual agreement of the initiator and responder (see Section 4.3 and Section 6.2 of [RFC6891]), and it is recommended to support EDNS. The 512 octets UDP message size limit will remain in practical effect until substantially all DNS servers and resolvers support EDNS.

Since DNS responses include a copy of the request, the space available for response data is somewhat less than the full 512 octets. Negative responses are quite small, but for positive and referral responses, every octet must be carefully and sparingly allocated. While the response size of positive responses is also a concern in [RFC3226], this document specifically addresses referral response size.

While more than fourteen years passed since the publication of the original EDNS0 document [RFC2671], measurements conducted at the M Root Server in May 2012 suggested that only around 65% of initiators support it. This fraction was consistent with similar measurements conducted in 2010 and 2011. The long tail of EDNS deployment may eventually be measured in decades.

DNS initiators and responders that support DNSSEC [RFC4033], and signal a desire to use it, can expect larger response sizes in the case where those responses contain DNSSEC RRSets. EDNS support in DNSSEC-aware initiators and responders can be assumed, since the desire to use DNSSEC is signalled using the DO flag in the EDNS0 header.

Even in scenarios where EDNS support in initiators and responders can be assumed, e.g. in the case of messages exchanged using DNSSEC, or at some future time where EDNS deployment can be considered ubiquitous, there will still be cases when MTU limitations or IP fragmentation/reassembly problems in firewalls and other middleboxes will cause EDNS failures which lead to non-extended DNS retries. A smaller referral response will always be better than a larger one if the same end result can be achieved either way. See [RFC5625], [SAC035], and Section 6.2.6 of [RFC6891] for further discussion.

3. Delegation Details

3.1. Relevant Protocol Elements

A positive delegation response will include the following elements:

Section	Description
Header Section	Fixed length (12 octets)
Question Section	Original query (name, class, type)
Answer Section	Empty, or a CNAME/DNAME chain
Authority Section	NS RSet (name server names)
Additional Section	A and AAAA RSets (name server addresses)

If the total size of the UDP response exceeds 512 octets or the size advertised in EDNS, and if the data that does not fit was "required", then the TC bit will be set to indicate truncation. This will usually cause the requester to retry using TCP, depending on what information was desired and what information was omitted. For example, truncation in the authority section is of no interest to a stub resolver who only plans to consume the answer section. If a retry using TCP is needed, the total cost of the transaction is much higher. See Section 6.1.3.2 of [RFC1123] for details on the requirement that UDP be attempted before falling back to TCP.

RRSets are never sent partially unless the TC bit is set to indicate truncation. When the TC bit is set, the final apparent RSet in the final non-empty section must be considered "possibly damaged" (see Section 6.2 of [RFC1035] and Section 9 of [RFC2181]).

With or without truncation, the glue present in the additional data section should be considered "possibly incomplete", and requesters should be prepared to re-query for any damaged or missing RSets. Note that truncation of the additional data section might not be signaled via the TC bit since additional data is often optional (see discussion in Appendix B of [RFC4472]).

DNS label compression allows the component labels of a domain name to be instantiated exactly once per DNS message, and then referenced with a two-octet "pointer" from other locations in that same DNS message (see Section 4.1.4 of [RFC1035]). If all name server names in a message share a common parent domain (for example, all of them

are in the "ROOT-SERVERS.NET" domain), then more space will be available for incompressible data (such as name server addresses).

The query name can be as long as 255 octets of network data. In this worst case scenario, the question section will be 259 octets in size, which would leave only 240 octets for the authority and additional sections (after deducting 12 octets for the fixed length header) in a referral.

3.2. Advice to Zone Administrators

Average and maximum question section sizes can be predicted by the zone administrator, since they will know what names actually exist and can measure which ones are queried for most often. Note that if the zone contains any wildcards, it is possible for maximum length queries to require positive responses, but that it is reasonable to expect truncation and TCP retry in that case. For cost and performance reasons, the majority of requests should be satisfied without truncation or TCP retry.

Some queries for non-existent names can be large. If DNSSEC is not being used this is unlikely to pose a problem since unsigned negative responses need not contain any answer, authority or additional records. See Section 2.1 of [RFC2308] for more information about the format of negative responses without DNSSEC. Negative responses from DNSSEC-signed zones can be much larger, however, due to the need to provide authenticated denial of existence [RFC7129].

The minimum useful number of name servers is two, for redundancy (see Section 4.1 of [RFC1034]). A zone's name servers should be reachable by all IP protocols versions (e.g., IPv4 and IPv6) in common use. As long as the servers are well managed, the server serving IPv6 might be different from the server serving IPv4 sharing the same server name.

The best case is no truncation at all. This is because many requesters will retry using TCP immediately, or will automatically requery for RRSets that are possibly truncated, without considering whether the omitted data was actually necessary.

Anycast [RFC3258] [RFC4786] is a useful technique for improving performance and below the zone cut being described by a delegation is responses.

While it is irrelevant to the response size issue, all zones have to be served via IPv4 as well as IPv6 to avoid name space fragmentation [RFC3901].

3.3. Advice to Server Implementors

Each NS RR for a zone will add 12 fixed octets (name, type, class, ttl, and rdlen) plus 2 to 255 variable octets (for the NSDNAME). Each A RR will require 16 octets, and each AAAA RR will require 28 octets.

While DNS distinguishes between necessary and optional resource records, this distinction is according to protocol elements necessary to signify facts, and takes no official notice of protocol content necessary to ensure correct operation. For example, a name server name that is in or below the zone cut being described by a delegation is "necessary content", since there is no way to reach that zone unless the parent zone's delegation includes "glue records" describing that name server's addresses.

Recall that the TC bit is only set when a required RRSet can not be included in its entirety (see Section 9 of [RFC2181]). Even when some of the RRSets to be included in the additional section don't fit in the response size, the TC bit isn't set. These RRSets may be important for a referral. Some DNS implementations try to resolve these missing glue records separately which will introduce extra queries and extra time to resolve a given name.

A delegation response should prioritize glue records as follows.

first: All glue RRSets for one name server whose name is in or below the zone being delegated, or which has multiple address RRSets (currently A and AAAA), or preferably both;

second: Alternate between adding all glue RRSets for any name servers whose names are in or below the zone being delegated, and all glue RRSets for any name servers who have multiple address RRSets (currently A and AAAA);

thence: All other glue RRSets, in any order.

Whenever there are multiple candidates for a position in this priority scheme, one should be chosen on a round-robin or fully random basis. The goal of this priority scheme is to offer "necessary" glue first to fill into the response if possible.

If any "necessary" content cannot be fit in the response, then it is advisable that the TC bit be set in order to force a TCP retry, rather than have the zone be unreachable. Note that a parent server's proper response to a query for in-child glue or below-child glue is a referral rather than an answer, and that this referral must be able to contain the in-child or below-child glue, and that in

outlying cases, only EDNS or TCP will be large enough to contain that data.

The glue record order should be independent of the version of IP used in the query because the DNS server might just see a query from an intermediate server rather than the query from the original client.

4. Analysis

An instrumented protocol trace of a best case delegation response is shown in Figure 1. Note that 13 servers are named, and 13 addresses are given. This query was artificially designed to exactly reach the 512 octets limit.

```
;; flags: qr rd; QUERY: 1, ANS: 0, AUTH: 13, ADDIT: 13
;; QUERY SECTION:
;; [23456789.123456789.123456789.\
    123456789.123456789.123456789.com A IN]           ;; @80

;; AUTHORITY SECTION:
com.          172800 NS  E.GTLD-SERVERS.NET.      ;; @112
com.          172800 NS  F.GTLD-SERVERS.NET.      ;; @128
com.          172800 NS  G.GTLD-SERVERS.NET.      ;; @144
com.          172800 NS  H.GTLD-SERVERS.NET.      ;; @160
com.          172800 NS  I.GTLD-SERVERS.NET.      ;; @176
com.          172800 NS  J.GTLD-SERVERS.NET.      ;; @192
com.          172800 NS  K.GTLD-SERVERS.NET.      ;; @208
com.          172800 NS  L.GTLD-SERVERS.NET.      ;; @224
com.          172800 NS  M.GTLD-SERVERS.NET.      ;; @240
com.          172800 NS  A.GTLD-SERVERS.NET.      ;; @256
com.          172800 NS  B.GTLD-SERVERS.NET.      ;; @272
com.          172800 NS  C.GTLD-SERVERS.NET.      ;; @288
com.          172800 NS  D.GTLD-SERVERS.NET.      ;; @304

;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET. 172800 A   192.5.6.30          ;; @320
B.GTLD-SERVERS.NET. 172800 A   192.33.14.30         ;; @336
C.GTLD-SERVERS.NET. 172800 A   192.26.92.30         ;; @352
D.GTLD-SERVERS.NET. 172800 A   192.31.80.30         ;; @368
E.GTLD-SERVERS.NET. 172800 A   192.12.94.30         ;; @384
F.GTLD-SERVERS.NET. 172800 A   192.35.51.30         ;; @400
G.GTLD-SERVERS.NET. 172800 A   192.42.93.30         ;; @416
H.GTLD-SERVERS.NET. 172800 A   192.54.112.30        ;; @432
I.GTLD-SERVERS.NET. 172800 A   192.43.172.30        ;; @448
J.GTLD-SERVERS.NET. 172800 A   192.48.79.30         ;; @464
K.GTLD-SERVERS.NET. 172800 A   192.52.178.30        ;; @480
L.GTLD-SERVERS.NET. 172800 A   192.41.162.30        ;; @496
M.GTLD-SERVERS.NET. 172800 A   192.55.83.30         ;; @512

;; MSG SIZE  sent: 80  rcvd: 512
```

Figure 1

For longer query names, the number of address records supplied will be lower. Furthermore, it is only by using a common parent name (which is "GTLD-SERVERS.NET." in this example) that all 13 addresses are able to fit, due to the use of label compression pointers in the last 12 occurrences of the parent domain name. The outputs from the response simulator in Appendix A (written in perl [PERL]) shown in Figure 2 and Figure 3 demonstrate these properties.

```
% perl respsize.pl a.dns.br b.dns.br c.dns.br d.dns.br
a.dns.br requires 10 bytes
b.dns.br requires 4 bytes
c.dns.br requires 4 bytes
d.dns.br requires 4 bytes
# of NS: 4
For maximum size query (255 byte):
  only A is considered:      # of A is 4 (green)
  A and AAAA are considered: # of A+AAAA is 3 (yellow)
  preferred-glue A is assumed: # of A is 4, # of AAAA is 3 (yellow)
For average size query (64 byte):
  only A is considered:      # of A is 4 (green)
  A and AAAA are considered: # of A+AAAA is 4 (green)
  preferred-glue A is assumed: # of A is 4, # of AAAA is 4 (green)
```

Figure 2

```
% perl respsize.pl ns-ext.isc.org ns.psg.com ns.ripe.net ns.eu.int
ns-ext.isc.org requires 16 bytes
ns.psg.com requires 12 bytes
ns.ripe.net requires 13 bytes
ns.eu.int requires 11 bytes
# of NS: 4
For maximum size query (255 byte):
  only A is considered:      # of A is 4 (green)
  A and AAAA are considered: # of A+AAAA is 3 (yellow)
  preferred-glue A is assumed: # of A is 4, # of AAAA is 2 (yellow)
For average size query (64 byte):
  only A is considered:      # of A is 4 (green)
  A and AAAA are considered: # of A+AAAA is 4 (green)
  preferred-glue A is assumed: # of A is 4, # of AAAA is 4 (green)
```

Figure 3

Here we use the term "green" if all address records could fit, or "yellow" if two or more could fit, or "orange" if only one could fit, or "red" if no address record could fit. It's clear that without a

common parent for name server names, much space would be lost. For these examples we use an average/common name size of 15 octets, befitting our assumption of "GTLD-SERVERS.NET." as our common parent name.

We assume a medium query name size of 64 since that is the typical size seen in trace data at the time of this writing. If Internationalized Domain Name (IDN) or any other technology that results in larger query names be deployed significantly in advance of EDNS, then new measurements and new estimates will have to be made.

5. Conclusions

The current practice of giving all name server names a common parent (such as "GTLD-SERVERS.NET." or "ROOT-SERVERS.NET.") saves space in DNS responses and allows for more name servers to be enumerated than would otherwise be possible, since the common parent domain name only appears once in a DNS message and is referred to via "compression pointers" thereafter.

If all name server names for a zone share a common parent, then it is operationally advisable to make all servers for the zone thus served also be authoritative for the zone of that common parent. For example, the root name servers (?.ROOT-SERVERS.NET.) can answer authoritatively for the ROOT-SERVERS.NET. zone. This is to ensure that the zone's servers always have the zone's name servers' glue available when delegating, and will be able to respond with answers rather than referrals if a requester who wants that glue comes back asking for it. In this case the name server will likely be a "stealth master" -- authoritative but not advertised in the glue zone's NS RRSet. See Section 2 of [RFC1996] for more information about stealth masters.

Thirteen (13) is the effective maximum number of name server names usable with traditional (non-extended) DNS, assuming a common parent domain name, and given that implicit referral response truncation is undesirable in the average case.

More than one address record in a protocol family per server is inadvisable since the necessary glue RRSets (A or AAAA) are atomically indivisible, and will be larger than a single resource record. Larger RRSets are more likely to lead to or encounter truncation.

More than one address record across protocol families is less likely to lead to or encounter truncation, partly because multiprotocol clients, which are required to handle larger RRSets such as AAAA RRs, are more likely to speak EDNS, which can use a larger UDP response size limit, and partly because the resource records (A and AAAA) are in different RRSets and are therefore divisible from each other.

Name server names that are at or below the zone they serve are more sensitive to referral response truncation, and glue records for them should be considered "more important" than other glue records, in the assembly of referral responses.

6. Security Considerations

The recommendations contained in this document have no known security implications.

7. IANA Considerations

This document has no IANA actions.

8. Acknowledgements

The authors thank Peter Koch, Rob Austein, Mark Andrews, Kenji Rikitake, Stephane Bortzmeyer, Olafur Gudmundsson, Alfred Hoenes, Alexander Mayrhofer, and Ray Bellis for their valuable comments and suggestions.

This work was supported by the US National Science Foundation (research grant SCI-0427144) and DNS-OARC.

9. References

9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

9.2. Informative References

- [PERL] Wall, L., Christiansen, T., and J. Orwant, "Programming Perl, 3rd ed.", ISBN 0-596-00027-8, July 2000.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.
- [RFC2672] Crawford, M., "Non-Terminal DNS Name Redirection", RFC 2672, August 1999.
- [RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver message size requirements", RFC 3226, December 2001.
- [RFC3258] Hardie, T., "Distributing Authoritative Name Servers via Shared Unicast Addresses", RFC 3258, April 2002.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

- [RFC3901] Durand, A. and J. Ihren, "DNS IPv6 Transport Operational Guidelines", BCP 91, RFC 3901, September 2004.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC4472] Durand, A., Ihren, J., and P. Savola, "Operational Considerations and Issues with IPv6 DNS", RFC 4472, April 2006.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, December 2006.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, August 2009.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.
- [RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, February 2014.
- [SAC035] Bellis, R. and L. Phifer, "Test Report: DNSSEC Impact on Broadband Routers and Firewalls", SAC 035, September 2008.

Appendix A. The response simulator program

```
#!/usr/bin/perl
#
# SYNOPSIS
#   respsize.pl [ -z zone ] fqdn_ns1 fqdn_ns2 ...
#       if all queries are assumed to have a same zone suffix,
#       such as "jp" in JP TLD servers, specify it in -z option
#
use strict;
use Getopt::Std;

my ($sz_msg) = (512);
my ($sz_header, $sz_ptr, $sz_rr_a, $sz_rr_aaaa) = (12, 2, 16, 28);
my ($sz_type, $sz_class, $sz_ttl, $sz_rdlenn) = (2, 2, 4, 2);
my (%namedb, $name, $nssect, %opts, $optz);
my $n_ns = 0;

getopt('z', %opts);
if (defined($opts{'z'})) {
    server_name_len($opts{'z'}); # just register it
}

foreach $name (@ARGV) {
    my $len;
    $n_ns++;
    $len = server_name_len($name);
    print "$name requires $len bytes\n";
    $nssect += $sz_ptr + $sz_type + $sz_class + $sz_ttl
        + $sz_rdlenn + $len;
}
print "# of NS: $n_ns\n";
arsect(255, $nssect, $n_ns, "maximum");
arsect(64, $nssect, $n_ns, "average");

sub server_name_len {
    my ($name) = @_;
    my (@labels, $len, $n, $suffix);

    $name =~ tr/A-Z/a-z/;
    @labels = split(/\./, $name);
    $len = length(join('.', @labels)) + 2;
    for ($n = 0; $#labels >= 0; $n++, shift @labels) {
        $suffix = join('.', @labels);
        return length($name) - length($suffix) + $sz_ptr
            if (defined($namedb{$suffix}));
        $namedb{$suffix} = 1;
    }
}
```

```

    return $len;
}

sub arsect {
    my ($sz_query, $nssect, $n_ns, $cond) = @_ ;
    my ($space, $n_a, $n_a_aaaa, $n_p_aaaa, $ansect);
    $ansect = $sz_query + $sz_type + $sz_class;
    $space = $sz_msg - $sz_header - $ansect - $nssect;
    $n_a = atmost(int($space / $sz_rr_a), $n_ns);
    $n_a_aaaa = atmost(int($space
                           / ($sz_rr_a + $sz_rr_aaaa)), $n_ns);
    $n_p_aaaa = atmost(int(($space - $sz_rr_a * $n_ns)
                          / $sz_rr_aaaa), $n_ns);
    printf "For %s size query (%d byte):\n", $cond, $sz_query;
    printf "    only A is considered:      ";
    printf "# of A is %d (%s)\n", $n_a, &judge($n_a, $n_ns);
    printf "    A and AAAA are considered:  ";
    printf "# of A+AAAA is %d (%s)\n",
           $n_a_aaaa, &judge($n_a_aaaa, $n_ns);
    printf "    preferred-glue A is assumed: ";
    printf "# of A is %d, # of AAAA is %d (%s)\n",
           $n_a, $n_p_aaaa, &judge($n_p_aaaa, $n_ns);
}

sub judge {
    my ($n, $n_ns) = @_ ;
    return "green" if ($n >= $n_ns);
    return "yellow" if ($n >= 2);
    return "orange" if ($n == 1);
    return "red";
}

sub atmost {
    my ($a, $b) = @_ ;
    return 0 if ($a < 0);
    return $b if ($a > $b);
    return $a;
}

```

Appendix B. Editorial Notes

This section (and sub-sections) to be removed prior to publication.

B.1. Change History

- 15 Draft resurrected; Joe added as co-author; changed Paul's affiliation. Minor wordsmithing to account for the passage of time. Terminology section added. Added commentary on DNSSEC impact on response sizes and EDNS support.

Authors' Addresses

Paul Vixie
Farsight Security, Inc.
155 Bovet Road, #476
San Mateo, CA 94402
USA

Phone: +1 650 489 7919
Email: vixie@farsightsecurity.com

Akira Kato
Keio University/WIDE Project
Graduate School of Media Design
4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
Japan

Phone: +81 45 564 2490
Email: kato@wide.ad.jp

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

Network Working Group
Internet-Draft
Obsoletes: RFC6304 (if approved)
Intended status: Informational
Expires: August 17, 2014

J. Abley
Dyn, Inc.
W. Maton
DNS-OARC
February 13, 2014

AS112 Nameserver Operations
draft-jabley-dnsop-rfc6304bis-00

Abstract

Many sites connected to the Internet make use of IPv4 addresses that are not globally-unique. Examples are the addresses designated in RFC 1918 for private use within individual sites.

Devices in such environments may occasionally originate Domain Name System (DNS) queries (so-called "reverse lookups") corresponding to those private-use addresses. Since the addresses concerned have only local significance, it is good practice for site administrators to ensure that such queries are answered locally. However, it is not uncommon for such queries to follow the normal delegation path in the public DNS instead of being answered within the site.

It is not possible for public DNS servers to give useful answers to such queries. In addition, due to the wide deployment of private-use addresses and the continuing growth of the Internet, the volume of such queries is large and growing. The AS112 project aims to provide a distributed sink for such queries in order to reduce the load on the corresponding authoritative servers. The AS112 project is named after the Autonomous System Number (ASN) that was assigned to it.

RFC6304 described the steps required to install a new AS112 node, and offered advice relating to such a node's operation. This document updates that advice to facilitate the addition and removal of zones for which query traffic will be sunk at AS112 nodes, using DNAME, whilst still supporting direct delegations to AS112 name servers.

This document obsoletes RFC6304.

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 August 17, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. AS112 DNS Service	5
2.1. Approach	5
2.1.1. Direct Delegation	5
2.1.2. DNAME Redirection	5
2.2. Zones	5
2.3. Nameservers	6
3. Installation of a New Node	7
3.1. Useful Background Knowledge	7
3.2. Topological Location	7
3.3. Operating System and Host Considerations	7
3.4. Routing Software	8
3.5. DNS Software	10
3.6. Testing a Newly-Installed Node	14
4. Operations	16
4.1. Monitoring	16
4.2. Downtime	16
4.3. Statistics and Measurement	16
5. Communications	17
6. On the Future of AS112 Nodes	18
7. IANA Considerations	19
8. Security Considerations	20
9. Acknowledgements	21
10. References	22
10.1. Normative References	22
10.2. Informative References	22
Appendix A. History	25
Appendix B. Revision History and Venue	26
B.1. draft-jabley-dnsop-rfc6304bis-00	26
Authors' Addresses	27

1. Introduction

Many sites connected to the Internet make use of IPv4 addresses that are not globally unique. Examples are the addresses designated in [RFC1918] for private use within individual sites.

Devices in such environments may occasionally originate Domain Name System (DNS) [RFC1034] queries (so-called "reverse lookups") corresponding to those private-use addresses. Since the addresses concerned have only local significance, it is good practice for site administrators to ensure that such queries are answered locally [RFC6303]. However, it is not uncommon for such queries to follow the normal delegation path in the public DNS instead of being answered within the site.

It is not possible for public DNS servers to give useful answers to such queries. In addition, due to the wide deployment of private-use addresses and the continuing growth of the Internet, the volume of such queries is large and growing. The AS112 project aims to provide a distributed sink for such queries in order to reduce the load on the IN-ADDR.ARPA authoritative servers [RFC5855].

The AS112 project encompasses a loosely coordinated collection of independently operated name servers. Each name server functions as a single node in an AS112 anycast cloud [RFC4786], and is configured to answer authoritatively for a particular set of nominated zones.

The AS112 project is named after the Autonomous System Number (ASN) that was assigned to it (see Appendix A).

2. AS112 DNS Service

2.1. Approach

2.1.1. Direct Delegation

[RFC6304] describes an approach whereby zones whose traffic should be directed towards an AS112 sink should be directly delegated to AS112 name servers. Correspondingly, each AS112 node is manually configured to answer appropriately for those zones.

The guidance in this document preserves this capability for the zones that were originally delegated in this fashion. AS112 nodes that were implemented in accordance with the guidance in [RFC6304] will continue to provide service for those zones.

2.1.2. DNAME Redirection

[I-D.ietf-dnsop-as112-dname] describes a different approach whereby queries towards specific zones are redirected to an empty zone also hosted on AS112 servers, using DNAME [RFC6672].

The guidance in this document introduces this capability, allowing any zone administrator to sink query traffic in AS112 infrastructure without requiring changes to any AS112 node.

2.2. Zones

To support Direct Delegation AS112 service, AS112 name servers answer authoritatively for the following zones, corresponding to [RFC1918] private-use netblocks:

- o 10.IN-ADDR.ARPA
- o 16.172.IN-ADDR.ARPA, 17.172.IN-ADDR.ARPA, ..., 31.172.IN-ADDR.ARPA
- o 168.192.IN-ADDR.ARPA

and the following zone, corresponding to the "link local" netblock 169.254.0.0/16 listed in [RFC5735]:

- o 254.169.IN-ADDR.ARPA

To support DNAME Redirection AS112 service, AS112 name servers answer authoritatively for the following zone, as specified in [I-D.ietf-dnsop-as112-dname]:

- o EMPTY.AS112.ARPA

To aid identification of AS112 anycast nodes, each node also answers authoritatively for the following zones:

- o HOSTNAME.AS112.NET
- o HOSTNAME.AS112.ARPA

See Section 3.5 for the recommended contents of all these zones.

2.3. Nameservers

To support Direct Delegation AS112 service, the relevant zones listed in Section 2.2 are delegated to the two name servers BLACKHOLE-1.IANA.ORG (192.175.48.6) and BLACKHOLE-2.IANA.ORG (192.175.48.42).

Additionally, the server PRISONER.IANA.ORG (192.175.48.1) is listed in the MNAME field of the SOA records of the IN-ADDR.ARPA zones served by AS112 name servers. PRISONER.IANA.ORG receives mainly dynamic update queries.

The addresses of all these name servers are covered by the single IPv4 prefix 192.175.48.0/24.

To support DNAME Redirection AS112 service, the single zone EMPTY.AS112.ARPA is delegated to the single name server BLACKHOLE.AS112.ARPA (addresses TBA). The addresses of that name server are covered by the single IPv4 prefix TBA, and the single IPv6 prefix TBA.

3. Installation of a New Node

3.1. Useful Background Knowledge

Installation of an AS112 node is relatively straightforward. However, experience in the following general areas may prove useful:

- o inter-domain routing with BGP [RFC4271];
- o DNS authoritative server operations;
- o anycast [RFC4786] distribution of DNS services.

3.2. Topological Location

AS112 nodes may be located anywhere on the Internet. For nodes that are intended to provide a public service to the Internet community (as opposed to private use), it may well be advantageous to choose a location that is easily (and cheaply) reachable by multiple providers, such as an Internet exchange point.

AS112 nodes may advertise their service prefix to BGP peers for local use (analogous to a conventional peering relationship between two providers) or for global use (analogous to a customer relationship with one or more providers).

It is good operational practice to notify the community of users that may fall within the reach of a new AS112 node before it is installed. At an Internet Exchange, local mailing lists usually exist to facilitate such announcements. For nodes that are intended to be globally reachable, coordination with other AS112 operators is highly recommended. See also Section 5.

3.3. Operating System and Host Considerations

Examples in this document are based on UNIX and UNIX-like operating systems, but other operating systems exist which are suitable for use in construction of an AS112 node.

The chosen platform should include support for either cloned loopback interfaces, or the capability to bind multiple addresses to a single loopback interface. The addresses of the name servers listed in Section 2.3 will be configured on these interfaces in order that the DNS software can respond to queries properly.

A host that is configured to act as an AS112 anycast node should be dedicated to that purpose, and should not be used to simultaneously provide other services. This guidance is provided due to the

unpredictable (and occasionally high) traffic levels that AS112 nodes have been seen to attract.

System startup scripts should be arranged such that the various AS112-related components start automatically following a system reboot. The order in which interfaces are configured and software components started should be arranged such that routing software startup follows DNS software startup, and DNS software startup follows loopback interface configuration.

Wrapper scripts or other arrangements should be employed to ensure that the anycast service prefix for AS112 is not advertised while either the anycast addresses are not configured, or while the DNS software is not running.

3.4. Routing Software

AS112 nodes signal the availability of AS112 name servers to the Internet using BGP [RFC4271]: each AS112 node is a BGP speaker, and announces the prefix 192.175.48.0/24 to the Internet with origin AS 112 (see also Section 2.3).

The examples in this document are based on the Quagga Routing Suite [1] running on Linux, but other software packages exist which also provide suitable BGP support for AS112 nodes.

The "bgpd.conf" file is used by Quagga's bgpd daemon, which provides BGP protocol support. The router id in this example is 203.0.113.1; the AS112 node peers with external peers 192.0.2.1 and 192.0.2.2. Note the local AS number 112, and the origination of the prefix 192.175.48.0/24 to support Direct Delegation AS112 service; the IPv4 prefix TBA and the IPv6 prefix TBA support DNAME Redirection.

```
! bgpd.conf
!
hostname as112-bgpd
password <something>
enable password <supersomething>
!
! Note that all AS112 nodes use the local Autonomous System
! Number 112, and originate the IPv4 prefixes 192.175.48.0/24
! and TBA and the IPv6 prefix TBA.
!
! All other addresses shown below are illustrative, and
! actual numbers will depend on local circumstances.
!
! IPv4-only or IPv6-only AS112 nodes should omit advertisements
! for address families they do not support.
!
router bgp 112
  bgp router-id 203.0.113.1
  neighbor 192.0.2.1 remote-as 64496
  neighbor 192.0.2.1 next-hop-self
  neighbor 192.0.2.2 remote-as 64497
  neighbor 192.0.2.2 next-hop-self
!
  address-family ipv4 unicast
    network 192.175.48.0/24
    network TBA
!
  address-family ipv6 unicast
    network TBA
```

The "zebra.conf" file is required to provide integration between protocol daemons (bgpd, in this case) and the kernel.

```
! zebra.conf
!
hostname as112
password <something>
enable password <supersomething>
!
interface lo
!
interface eth0
!
```

3.5. DNS Software

Although the queries received by AS112 nodes are definitively misdirected, it is important that they be answered in a manner that is accurate and consistent. For this reason AS112 nodes operate as fully-functional and standards-compliant DNS authoritative servers [RFC1034], and hence require appropriate DNS software.

Examples in this document are based on ISC BIND9 [2], but other DNS software exists which is suitable for use in construction of an AS112 node.

The following is a sample BIND9 "named.conf" file for a dedicated AS112 server. Note that the name server is configured to act as an authoritative-only server (i.e. recursion is disabled). The name server is also configured to listen on the various AS112 anycast name server addresses, as well as its local addresses.

```
// named.conf

// global options

options {
    listen-on {
        127.0.0.1;           // localhost

        // the following address is node-dependent, and should be set to
        // something appropriate for the new AS112 node

        203.0.113.1;        // local address (globally-unique, unicast)

        // the following addresses are used to support Direct Delegation
        // AS112 service, and are the same for all AS112 nodes

        192.175.48.1;        // prisoner.iana.org (anycast)
        192.175.48.6;        // blackhole-1.iana.org (anycast)
        192.175.48.42;       // blackhole-2.iana.org (anycast)

        // the following address is used to support DNAME Redirection
        // AS112 service, and is the same for all AS112 nodes

        TBA;                 // blackhole.as112.arpa (anycast)
    };

    listen-on-v6 {
        ::1;                 // localhost

        // the following address is used to support DNAME Redirection
```

```
// AS112 service, and is the same for all AS112 nodes

    TBA;                // blackhole.as112.arpa (anycast)
};

directory "/var/named";
recursion no;          // authoritative-only server
query-source address *;
};

// log queries, so that when people call us about unexpected
// answers to queries they didn't realise they had sent, we
// have something to talk about. Note that activating this
// has the potential to create high CPU load and consume
// enormous amounts of disk space.

logging {
    channel "querylog" {
        file "/var/log/query.log" versions 2 size 500m;
        print-time yes;
    };
    category queries { querylog; };
};

// Direct Delegation AS112 Service

// RFC 1918

zone "10.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "16.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "17.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "18.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "19.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "20.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "21.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "22.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "23.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "24.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "25.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "26.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "27.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "28.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "29.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "30.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "31.172.in-addr.arpa" { type master; file "db.dd-empty"; };
zone "168.192.in-addr.arpa" { type master; file "db.dd-empty"; };

// RFC 5735
```

```
zone "254.169.in-addr.arpa" { type master; file "db.dd-empty"; };

// DNAME Redirection AS112 Service

zone "empty.as112.arpa" { type master; file "db.dr-empty"; };

// also answer authoritatively for the HOSTNAME.AS112.NET and
// HOSTNAME.AS112.ARPA zones, which contain data of operational
// relevance

zone "hostname.as112.net" {
    type master;
    file "db.hostname.as112.net";
};

zone "hostname.as112.arpa" {
    type master;
    file "db.hostname.as112.arpa";
};
```

The "db.dd-empty" file follows, below. This is the source data used to populate all the IN-ADDR.ARPA zones listed in Section 2.2 that support Direct Delegation AS112 service. Note that the RNAME specified in the SOA record corresponds to `hostmaster@root-servers.org`, a suitable e-mail address for technical queries about these zones.

```
; db.dd-empty
;
; Empty zone for Direct Delegation AS112 service.
;
$TTL      1W
@ IN SOA  prisoner.iana.org. hostmaster.root-servers.org. (
                                1          ; serial number
                                1W         ; refresh
                                1M         ; retry
                                1W         ; expire
                                1W )       ; negative caching TTL
;
    NS      blackhole-1.iana.org.
    NS      blackhole-2.iana.org.
;
; There should be no other resource records included in this zone.
;
; Records that relate to RFC 1918-numbered resources within the
; site hosting this AS112 node should not be hosted on this
; name server.
```

The "db.dr-empty" file follows, below. This is the source data used to populate the EMPTY.AS112.ARPA zone that supports DNAME Redirection AS112 service. Note that the RNAME specified in the SOA record corresponds to noc@dns.icann.org, a suitable e-mail address for technical queries about this zone.

```
; db.dr-empty
;
; Empty zone for Direct Delegation AS112 service.
;
$TTL      1W
@ IN SOA  blackhole.as112.arpa. noc.dns.icann.org. (
                                1          ; serial number
                                1W         ; refresh
                                1M         ; retry
                                1W         ; expire
                                1W )      ; negative caching TTL
;
      NS      blackhole.as112.arpa.
;
; There should be no other resource records included in this zone.
;
; Records that relate to RFC 1918-numbered resources within the
; site hosting this AS112 node should not be hosted on this
; name server.
```

The "db.hostname.as112.net" and "db.hostname.as112.arpa" files follow, below. These zones contain various resource records that provide operational data to users for troubleshooting or measurement purposes, and should be edited to suit local circumstances. Note that the responses to the queries "HOSTNAME.AS112.NET IN TXT" and "HOSTNAME.AS112.ARPA IN TXT" should fit within a 512 octet DNS/UDP datagram: i.e. it should be available over UDP transport without requiring EDNS0 support.

The optional LOC record [RFC1876] included in each zone apex provides information about the geospatial location of the node.

```

; db.hostname.as112.net
;
$TTL      1W
@         SOA      server.example.net. admin.example.net. (
                        1              ; serial number
                        1W             ; refresh
                        1M             ; retry
                        1W             ; expire
                        1W )           ; negative caching TTL
;
      NS      blackhole-2.iana.org.
      NS      blackhole-1.iana.org.
;
      TXT     "Name of Facility or similar" "City, Country"
      TXT     "See http://www.as112.net/ for more information."
;
      LOC     45 25 0.000 N 75 42 0.000 W 80.00m 1m 10000m 10m

; db.hostname.as112.arpa
;
$TTL      1W
@         SOA      server.example.net. admin.example.net. (
                        1              ; serial number
                        1W             ; refresh
                        1M             ; retry
                        1W             ; expire
                        1W )           ; negative caching TTL
;
      NS      blackhole.as112.arpa.
;
      TXT     "Name of Facility or similar" "City, Country"
      TXT     "See http://www.as112.net/ for more information."
;
      LOC     45 25 0.000 N 75 42 0.000 W 80.00m 1m 10000m 10m

```

3.6. Testing a Newly-Installed Node

The BIND9 tool "dig" can be used to retrieve the TXT resource records associated with the names "HOSTNAME.AS112.NET" and "HOSTNAME.AS112.ARPA", directed at one of the AS112 anycast name server addresses. Continuing the example from above, the response received should indicate the identity of the AS112 node that responded to the query. See Section 3.5 for more details about the resource records associated with "HOSTNAME.AS112.NET".

```
% dig @prisoner.iana.org hostname.as112.net txt +short +norec
"Name of Facility or similar" "City, Country"
"See http://www.as112.net/ for more information."
%
```

If the response received indicates a different node is being used, then there is probably a routing problem to solve. If there is no response received at all, there might be host or name server problem. Judicious use of tools such as traceroute, and consultation of BGP looking glasses might be useful in troubleshooting.

Note that an appropriate set of tests for a new server will include queries sent from many different places within the expected service area of the node, using both UDP and TCP transport, and exercising all three AS112 anycast name server addresses.

4. Operations

4.1. Monitoring

AS112 nodes should be monitored to ensure they are functioning correctly, just as with any other production service. An AS112 node that stops answering queries correctly can cause failures and timeouts in unexpected places and can lead to failures in dependent systems that can be difficult to troubleshoot.

4.2. Downtime

An AS112 node that needs to go off-line (e.g. for planned maintenance or as part of the diagnosis of some problem) should stop advertising the AS112 service prefix to its BGP peers. This can be done by shutting down the routing software on the node altogether or by causing the routing system to withdraw the route.

Withdrawing the service prefix is important in order to avoid blackholing query traffic in the event that the DNS software on the node is not functioning normally.

4.3. Statistics and Measurement

Use of the AS112 node should be measured in order to track long-term trends, identify anomalous conditions, and to ensure that the configuration of the AS112 node is sufficient to handle the query load.

Examples of free monitoring tools that might be useful to operators of AS112 nodes include:

- o bindgraph [3]
- o dnstop [4]
- o DSC [5]

5. Communications

It is good operational practice to notify the community of users that may fall within the reach of a new AS112 node before it is installed. At Internet Exchanges, local mailing lists usually exist to facilitate such announcements.

For nodes that are intended to be globally reachable, coordination with other AS112 operators is especially recommended. The mailing list <mailto:as112-ops@lists.dns-oarc.net> is operated for this purpose.

Information pertinent to AS112 operations is maintained at <<http://www.as112.net/>>.

Information about an AS112 node should also be published within the DNS, within the "HOSTNAME.AS112.NET" and "HOSTNAME.AS112.ARPA" zones. See Section 3.5 for more details.

6. On the Future of AS112 Nodes

It is recommended practice for the operators of recursive name servers to answer queries for zones served by AS112 nodes locally, such that queries never have an opportunity to reach AS112 servers [RFC6303]. Operational experience with AS112 nodes does not currently indicate an observable trend towards compliance with those recommendations, however.

It is expected that some DNS software vendors will include default configuration that will implement measures such as those described in [RFC6303]. If such software is widely deployed, it is reasonable to assume that the query load received by AS112 nodes will decrease; however, it is safe to assume that the query load will not decrease to zero, and consequently that AS112 nodes will continue to provide a useful service for the foreseeable future.

The use of DNAME Redirection to provide AS112 service is new, and hence is informed by minimal operational experience. The use of DNAME means that queries for many source zones could be redirected to AS112 infrastructure with no real opportunity for coordination.

If successful, and in the absence of operational concerns, the community might well recommend the retirement of the original Direct Delegation AS112 service, with equivalent service for the zones concerned instead provided using DNAME Redirection. This document makes no such recommendation, however.

7. IANA Considerations

The name servers associated with Direct Delegation AS112 service are all named under the domain IANA.ORG (see Section 2.3). However, the anycast infrastructure itself is operated by a loosely-coordinated, diverse mix of organisations across the Internet, and is not an IANA function.

The autonomous system number 112 and the IPv4 prefix 192.175.48.0/24 were assigned by ARIN.

The IPv4 prefix TBA and the IPv6 prefix TBA, used for DNAME Redirection AS112 service, were assigned by the IANA [I-D.ietf-dnsop-as112-dname].

This document has no IANA actions.

8. Security Considerations

Hosts should never normally send queries to AS112 servers; queries relating to private-use addresses should be answered locally within a site. Hosts that send queries to AS112 servers may well leak information relating to private infrastructure to the public network, and this could present a security risk. This risk is orthogonal to the presence or absence of authoritative servers for these zones in the public DNS infrastructure, however.

Queries that are answered by AS112 servers are usually unintentional; it follows that the responses from AS112 servers are usually unexpected. Unexpected inbound traffic can trigger intrusion detection systems or alerts by firewalls. Operators of AS112 servers should be prepared to be contacted by operators of remote infrastructure who believe their security has been violated. Advice to those who mistakenly believe that responses from AS112 nodes constitutes an attack on their infrastructure can be found in [RFC6305].

The deployment of AS112 nodes is very loosely coordinated compared to other services distributed using anycast. The malicious compromise of an AS112 node and subversion of the data served by the node is hence more difficult to detect due to the lack of central management. Since it is conceivable that changing the responses to queries received by AS112 nodes might influence the behaviour of the hosts sending the queries, such a compromise might be used as an attack vector against private infrastructure.

Operators of AS112 should take appropriate measures to ensure that AS112 nodes are appropriately protected from compromise, such as would normally be employed for production name server or network infrastructure. The guidance provided for root name servers in [RFC2870] may be instructive.

The zones hosted by AS112 servers are not signed with DNSSEC [RFC4033]. Given the distributed and loosely-coordinated structure of the AS112 service, the zones concerned could only be signed if the private key material used was effectively public, obviating any security benefit resulting from the use of those keys.

9. Acknowledgements

The authors wish to acknowledge the assistance of Bill Manning, John Brown, Marco D'Itri, Daniele Arena, Stephane Bortzmeyer, Frank Habicht, Chris Thompson, Peter Losher, Peter Koch, Alfred Hoenes and S. Moonesamy in the preparation of [RFC6304], which this document supercedes.

The authors further acknowledge the assistance of YOUR NAME HERE in the preparation of this document.

10. References

10.1. Normative References

- [I-D.ietf-dnsop-as112-dname]
Abley, J., Dickson, B., Kumari, W., and G. Michaelson,
"AS112 Redirection using DNAME",
draft-ietf-dnsop-as112-dname-00 (work in progress),
November 2013.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
STD 13, RFC 1034, November 1987.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
E. Lear, "Address Allocation for Private Internets",
BCP 5, RFC 1918, February 1996.
- [RFC2870] Bush, R., Karrenberg, D., Koster, M., and R. Plzak, "Root
Name Server Operational Requirements", BCP 40, RFC 2870,
June 2000.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S.
Rose, "DNS Security Introduction and Requirements",
RFC 4033, March 2005.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway
Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast
Services", BCP 126, RFC 4786, December 2006.

10.2. Informative References

- [RFC1876] Davis, C., Vixie, P., Goodwin, T., and I. Dickinson, "A
Means for Expressing Location Information in the Domain
Name System", RFC 1876, January 1996.
- [RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses",
RFC 5735, January 2010.
- [RFC5855] Abley, J. and T. Manderson, "Nameservers for IPv4 and IPv6
Reverse Zones", BCP 155, RFC 5855, May 2010.
- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163,
RFC 6303, July 2011.
- [RFC6304] Abley, J. and W. Maton, "AS112 Nameserver Operations",
RFC 6304, July 2011.

- [RFC6305] Abley, J. and W. Maton, "I'm Being Attacked by PRISONER.IANA.ORG!", RFC 6305, July 2011.
- [RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

URIs

- [1] <<http://www.quagga.net/>>
- [2] <<http://www.isc.org/software/BIND/>>
- [3] <<http://www.linux.it/~md/software/>>
- [4] <<http://dns.measurement-factory.com/tools/dnstop/>>
- [5] <<http://dns.measurement-factory.com/tools/dsc/>>

Appendix A. History

Widespread use of the private address blocks listed in [RFC1918] followed that document's publication in 1996. At that time the IN-ADDR.ARPA zone was served by root servers.

The idea of off-loading IN-ADDR.ARPA queries relating to [RFC1918] addresses from the root name servers was first proposed by Bill Manning and John Brown.

The use of anycast for distributing authoritative DNS service for [RFC1918] IN-ADDR.ARPA zones was subsequently proposed at a private meeting of root server operators.

ARIN provided an IPv4 prefix for the anycast service and also the autonomous system number 112 for use in originating that prefix. This assignment gave the project its name.

In 2002, the first AS112 anycast nodes were deployed.

In 2011, the IN-ADDR.ARPA zone was redelegated from the root servers to a new set of servers operated independently by AfrinIC, APNIC, ARIN, ICANN, LACNIC, and the RIPE NCC and named according to [RFC5855].

[RFC6304], the precursor to this document, was published in July 2011.

The use of anycast name servers in the AS112 project contributed to the operational experience of anycast DNS services, and it can be seen as a precursor to the anycast distribution of other authoritative DNS servers in subsequent years (e.g., various root servers).

Appendix B. Revision History and Venue

A suitable venue for discussion of this document is the dnsop working group. Private comments may also be directed at the authors.

This section (and sub-sections) should be removed prior to publication.

B.1. draft-jabley-dnsop-rfc6304bis-00

Initial revision of [RFC6304] intended to provide guidance consistent with [I-D.ietf-dnsop-as112-dname].

Authors' Addresses

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

William F. Maton Sotomayor
DNS Operations, Analysis and Research Centre
Email: wfms@dns-oarc.net

Domain Name System Operations
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2012

W. Mekking
NLnet Labs
July 8, 2011

DNSSEC Key Timing Considerations Follow-Up
draft-mekking-dnsop-dnssec-key-timing-bis-02

Abstract

This document describes issues surrounding the timing of events related to DNSSEC policy. It presents timelines for various key rollovers and DNSSEC policy changes regarding the key signing scheme. It explicitly identifies the relationships between the various parameters affecting the rollover process.

This document updates [draft-ietf-dnsop-dnssec-key-timing] [MM: If approved] as it covers timelines for key rollovers in more detail and it covers additional key rollover scenarios, including algorithm rollover and single type key rollovers.

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 January 9, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	Key Rollover Considerations	4
1.2.	Terminology	5
2.	Key Definitions	5
2.1.	Key Types	5
2.2.	Key States Unraveled	6
2.2.1.	Validation Components	6
2.2.1.1.	The Public Key Component	6
2.2.1.2.	The Signature Component	6
2.2.1.3.	The Secure Delegation Component	6
2.2.2.	Validation Component States	6
2.3.	Key States	8
2.4.	Key Goals	10
2.5.	Delay Timings	10
3.	Key Rollovers	11
3.1.	Key Rollover Stages	11
3.2.	ZSK Rollovers	13
3.2.1.	Double-Signature	13
3.2.2.	Pre-Publication	15
3.2.3.	Double-RRSIG	18
3.3.	KSK Rollovers	21
3.3.1.	Double-RRset	21
3.3.2.	Double-Signature	24
3.3.3.	Double-DS	26
3.3.4.	Interaction with Configured Trust Anchors	29
3.3.4.1.	Adding a KSK	29
3.3.4.2.	Removing a KSK	29
3.4.	Rollovers in a Single Type Signing Scheme	29
3.4.1.	Double-RRset	30
3.4.2.	Double-Signature	31
3.4.3.	Pre-Publication	32
3.4.4.	Double-DS	35
3.5.	Stand-by Keys	38
4.	Policy rollover	39
4.1.	Enabling DNSSEC	39
4.2.	Disabling DNSSEC	41
4.3.	Algorithm Rollover	42

4.4. KSK-ZSK Split or Single Type Signing Scheme	43
5. IANA Considerations	43
6. Security Considerations	43
7. Acknowledgements	43
8. Changelog	43
8.1. Changes with key-timing draft	43
8.2. From -00 to -01	44
8.3. From -01 to -02	44
9. References	45
9.1. Informative References	45
9.2. Normative References	45
Appendix A. List of Symbols	45

1. Introduction

DNS was not originally designed with security in mind. The Domain Name System Security Extensions (DNSSEC, [RFC4033], [RFC4034], [RFC4035]) add a security layer that provides data origin authentication and data integrity. A DNS zone that implements DNSSEC must have the ability to replace ("roll") keys. This will be needed for various reasons such as recovery from a key compromise, replacement of key-signing hardware (if used), or even just to implement a policy that requires periodic rollovers.

In addition, a DNS zone may be subject to a given DNSSEC Policy [dps-framework]. Normally, such a policy provides a methodology for key rollover. A DNS operator may choose to change the DNSSEC Policy for a zone, or switch to a different policy. Such a change may also trigger a key rollover scenario to occur.

Key rollovers are time critical, multiple steps processes. This document describes issues surrounding the timing of events in the rolling of DNSSEC keys.

The structure of this document is as follows. In section Section 2, more terminology on keys is provided. The timelines for the various methods of key rollovers are presented in section Section 3. Section Section 4 deals with key rollovers initiated by a change in a DNSSEC Policy.

[MM: Editorarial comments are indicated by square brackets and editor initials]

1.1. Key Rollover Considerations

A key rollover involves the replacement of active keys with new keys. In order to avoid the zone being seen as bogus during the transition, there are constraints on the times at which the keys are added to and removed from the zone. DNSSEC records are not only held at the authoritative name server, they are also cached at client validators. The data on these systems can be interlinked, meaning a validator may try to validate a signature retrieved from a cache with a key obtained separately. The rollover process needs to happen in such a way that at all times through the rollover the information is consistent.

There exist different flavours of key rollovers. When making a choice which type of rollover to implement, several considerations may be taken into account:

- o Size of the zone and the DNS response: Adding signatures increases the zone size and the size of DNS responses significantly. To keep the sizes of the zone and responses as small as possible, the DNSSEC records should be introduced only when they are required. For the same reason, dead keys and signatures must be removed periodically.
- o Size of the DNSKEY RRset: Instead of keeping the set of signatures to a minimum, it is also possible to minimize the size of the DNSKEY RRset. This consideration may be of importance in the case where trust anchor priming is an issue.
- o Interactions with the Parent: Where a key being replaced has a corresponding DS record in the parent zone, the rollover involves removing it and introducing the DS record corresponding to the new key. Such a process requires communication between the child and parent zones and may require additional operational work. This may lead to a sufficient delay. In the case where the interaction through the child-parent provisioning chain is unpredictable, it is preferred to keep the number of interactions with the parent to a minimum.

1.2. Terminology

The terminology used in this document is as defined in [RFC4033], [RFC4034], [RFC4035] and [RFC5011]. This document also introduces new terms in Section 2.

2. Key Definitions

2.1. Key Types

Keys can be used to authenticate information within the zone. Such keys are said to be Zone Signing Keys (ZSKs). In addition, keys can be used to authenticate the DNSKEY RRset in the zone. These keys are said to be Key Signing Keys (KSKs). Keys can be marked to be ZSK and KSK at the same time, for example in a Single Type Signing Scheme (STSS).

Despite that ZSK and KSK only describe the usage of a key, the terms are often used for identifying a key. However, when this document talks about a ZSK it actually means that the key is used as ZSK (but may also be used as KSK). In the same spirit, a KSK is a key that is used as KSK (but may also be used as ZSK). A key that is used as a KSK is responsible for creating a signature for the DNSKEY RRset. A key that is used as a ZSK is responsible for creating a signatures for all RRsets, except the DNSKEY RRset.

DNSSEC recognises the classification of keys with its SEP bit set and not set. Usually if a key is used as KSK, the SEP bit is set. However, a SEP bit setting has no effect on how a DNSKEY may be used. Policy determines whether the bit should be set, depending on the key's usage.

2.2. Key States Unraveled

In this document, the key states from [key-timing] have been unraveled. Instead of a single state, the state of all information associated with the key is represented separately. This information comprises up to three items called Validation Components: the public key, its created signatures, and the corresponding secure delegation.

2.2.1. Validation Components

2.2.1.1. The Public Key Component

The Public Key (DNSKEY) Component represents the state of the public part of the key. When talking about a KSK, this comprises the DNSKEY record and the RRSIG record for the DNSKEY RRset created with the key, as both the key and signature travel together. In the case of a ZSK, this comprises just the DNSKEY record.

[MM: Is this a safe assumption? Or are there rollover scenarios that benefit to decouple the DNSKEY RR and the RRSIG RR created with the key? For example in STSS environment.]

2.2.1.2. The Signature Component

The Signature (RRSIG) Component represents the state of the private part of the key. This comprises the RRSIG records for all RRsets excluding the DNSKEY RRset.

2.2.1.3. The Secure Delegation Component

The Secure Delegation (DS) Component represents the state of the secure delegation of the key. This comprises the DS record that corresponds to the Public Key Component.

2.2.2. Validation Component States

The consequence of this unraveling is that a single [key-timing] key state now comprises a set of multiple Validation Component states. A Validation Component may exist in up to two places: it can be present in the corresponding zone and it may be known in validator caches. Thus, all Validation Components follow the same state diagram:

Hidden --> Introduced --> Propagated --> Withdrawn --> Dead.

Hidden: The Validation Component is not available in the zone. In this state, no validators are able to fetch this Validation Component.

Introduced: The Validation Component is introduced and, as a result, is available in the zone. If the Validation Component comprises multiple RRs, the introduction may be done incrementally. As a result, the Validation Component that is said to be Introduced may be only partly available in the zone. In this state, there may be validators that fetch this Validation Component from the authoritative name server. However, there may also be validators that have associated information in the cache and don't use the new Validation Component.

Propagated: The Validation Component is available in the zone and enough time has passed to have it propagated into all validator caches. If the Validation Component comprises multiple RRs, it is said to be Propagated if and only if all RRs have been propagated into all validator caches. As a result, all validators fetch this Validation Component from cache or from the authoritative name server.

Withdrawn: The Validation Component is being withdrawn from the zone. If the Validation Component comprises multiple RRs, the withdrawal may be done incrementally. As a result, the Validation Component that is said to be Withdrawn may still be partly available in the zone. In this state, the Validation Component can also still live in validator caches.

Dead: The Validation Component is not available in the zone anymore and enough time has passed to have it expire from all validator caches.

A Key State can now be represented as the tuple (DNSKEY Component State, RRSIG Component State, DS Component State). For example:

$S(Kc) = (\text{DNSKEY Propagated}, \text{RRSIG Introduced}, \text{DS Hidden})$

where $S(Kc)$ is the state of the key (Kc), means that Kc is published in the zone and all the validators that have a copy of the DNSKEY RRset, have one that includes Kc . In addition, the key is being used for signing RRsets: RRSIG records made with Kc have been introduced in the zone. However, there may still be some validator caches that are unaware of these signatures. Finally, the corresponding DS record of Kc is said to be Hidden, meaning it has not yet been submitted to the parent.

For convenience, a ZSK can be represented as a tuple (DNSKEY State, RRSIG State), because the DS record is only used with KSKs. And a KSK can be represented as a tuple (DNSKEY State, DS State), because the RRSIG state only refers to ZSKs. The RRSIG record over the DNSKEY RRset should be published at the same time when the corresponding DNSKEY record is published. Therefore, both records will propagate to and expire from validator caches at the same time.

2.3. Key States

During the rolling process, a key moves through different states. Key States are derived from the Validation Component States. For example, if the DNSKEY Component of a key is in the Introduced State, the key is said to be Published. A key can be in multiple states at the same time.

Uninformed: A key is said to be Uninformed, if all Validation Components are in the Hidden state. The key has been created, but has not yet been used for anything.
 $S(k) = (\text{DNSKEY Hidden}, \text{RRSIG Hidden}, \text{DS Hidden})$

Published: A key is said to be Published if the DNSKEY Component is in the Introduced state. The DNSKEY record is published in the zone, but predecessors DNSKEY RRsets may be held in caches.
 $S(k) = (\text{DNSKEY Introduced}, \text{RRSIG } *, \text{DS } *)$

Active: A key is said to be Active if the RRSIG Component is in the Introduced state (for ZSKs).
 $S(k) = (\text{DNSKEY } *, \text{RRSIG Introduced}, \text{DS } *)$

ActiveDS: A key is said to be ActiveDS, or Submitted, if the DS Component is in the Introduced state (for KSKs).
 $S(k) = (\text{DNSKEY } *, \text{RRSIG } *, \text{DS Introduced})$

Known: A key is said to be Known if the DNSKEY Component is in the Propagated state. The new key data has been published for long enough to guarantee that any previous versions of it have expired from caches.
 $S(k) = (\text{DNSKEY Propagated}, \text{RRSIG } *, \text{DS } *)$

Safe: A key is said to be Safe if the RRSIG Component is in the Propagated state (for ZSKs). If a validator knows about the key, it is safe to assume that it may expect a signature created with this key.
 $S(k) = (\text{DNSKEY } *, \text{RRSIG Propagated}, \text{DS } *)$

SafeDS: A key is said to be SafeDS if the DS Component is in the Propagated state (for KSKs). If a validator knows about the key, it is safe to assume that it may expect a corresponding DS record for this key.

$S(k) = (\text{DNSKEY } *, \text{RRSIG } *, \text{DS Propagated})$

Removed: A key is said to be Removed if the DNSKEY Component is in the Withdrawn state. The key has been removed from the zone.

$S(k) = (\text{DNSKEY Withdrawn}, \text{RRSIG } *, \text{DS } *)$

Retired: A key is said to be Retired if the RRSIG Component is in the Withdrawn state (for ZSKs). Signatures are removed, or are incrementally being removed, from the zone. When a key is said to be Retired, there may still be caches that hold copies of the signatures.

$S(k) = (\text{DNSKEY } *, \text{RRSIG Withdrawn}, \text{DS } *)$

RetiredDS: A key is said to be RetiredDS if the DS Component is in the Withdrawn state (for KSKs). The request has been made to withdraw the DS record from the parent zone, but it may take some time before the record is actually removed. When a key is said to be RetiredDS, there may still be caches that hold copies of the DS record.

$S(k) = (\text{DNSKEY } *, \text{RRSIG } *, \text{DS Withdrawn})$

Forgotten: A key is said to be Forgotten if the DNSKEY Component is in the Dead state. At this point, no single validator cache should know about this key.

$S(k) = (\text{DNSKEY Dead}, \text{RRSIG } *, \text{DS } *)$

Expired: A key is said to be Expired if the RRSIG Component is in the Dead state (for ZSKs). At this point, no single validator cache should know about this key's signatures.

$S(k) = (\text{DNSKEY } *, \text{RRSIG Dead}, \text{DS } *)$

ExpiredDS: A key is said to be ExpiredDS if the DS Component is in the Dead state (for KSKs). At this point, no single validator cache should know about this key's DS record.

$S(k) = (\text{DNSKEY } *, \text{RRSIG } *, \text{DS Dead})$

Throughout the document, the states of a key K_c is denoted as $P(K_c)$. For example,

$P(K_c) = \text{Known Retired}$

means that key K_c is considered to be Known, the new key data has been published for long enough to guarantee that any previous versions of it have expired from caches, and it is considered to be

Retired, its signatures are being removed from the zone.

2.4. Key Goals

When performing a key rollover, it is usually intended to introduce a new key into the zone and to remove an existing key from the zone. These intentions can be called Key Goals. A Key Goal is the desire to make a key have certain Key State, as described in Section 2.3. During the lifetime of a key, the following goals may be put on a key:

- o Activate key: Make validators use the key's associated information to perform authentication. The goal is reached if the key is said to be Known and Safe(DS).
- o Remove key: Make validators forget about the key's associated information. The goal is reached if the key is said to be Forgotten and Expired(DS).
- o Stand-by key: Pre-publish information for this key to speed up a future (unscheduled) rollover. In case of a Stand-by ZSK, the goal is reached if the key is said to be Known. In case of a Stand-by KSK, the goal is reached if the key is said to be SafeDS.

2.5. Delay Timings

For every change made in the zone there are time delays that need to be taken into account:

Software Delay (Dsfw): The time it takes for the software to introduce the new information in the zone. This delay can vary a lot depending on the information that needs to be introduced. One can imagine that the software needs more time to sign a complete zone than when it pre-publishes a DNSKEY record. [MM: Dsfw maps to Dsgn from the key-timing draft]

Propagation Delay (Dprp): The time it takes for any change introduced at the master to replicate to all slave servers.

TTL Delay (Dttdl): The time it takes to expire the previous information from the validator caches. This delay depends on what RRsets need to expire from the caches. If not explicitly mentioned otherwise, Dttdl is considered the maximum TTL of the information that needs to expire from caches. Otherwise, Dttdl(RRtype) shows which specific RRsets need to expire. [MM: TTL terminology in key-timing draft: TTLds, TTLkey, TTLkeyC, TTLsoa, TTLsoaC, TTLsoaP, TTLsig]

Registration Delay to the Parent (Dreg): The time it takes to get the DS record to be placed into the parent zone, after it is submitted.

Propagation Delay of the Parent (DprpP): The time it takes for any change introduced at the parent master to replicate to all parent slave servers.

Despite the values of these delays may vary for the different rollover methods, the propagation delay to the caches can be defined as:

$D_{cacheZ} = D_{sfw} + D_{prp} + D_{ttl}$
 $D_{cacheK} = D_{sfw} + D_{prp} + D_{ttl}(DNSKEY)$
 $D_{cacheP} = D_{reg} + D_{prpP} + D_{ttl}(DS)$

where D_{cacheZ} is the propagation delay to the caches for information published in the zone, D_{cacheK} is the propagation delay to the caches for the DNSKEY RRset and D_{cacheP} is the propagation delay for information published in the parent zone.

[MM: Because some timings are unpredictable, it would make more sense to use triggering events instead of timings]

3. Key Rollovers

There are many different key rollover methods. Section 1.1 lists several considerations to prefer one method over the other. Though there are many different type of key rollovers, all methods share the same goal. There is a current key (K_c) that needs to be removed and a successor key (K_s) that needs to become active.

3.1. Key Rollover Stages

Broadly speaking, any key rollover can be thought of as the following sequence of stages:

Generation: In this stage, a new successor key is generated or derived from a key pool.

Preparation: In this stage, one or more Validation Components of the successor key are published in the zone and propagate through the nameserver network and into caches.

Ready: The Ready Stage always follows the Preparation Stage. The initial Validation Components of the successor key have been published long enough to guarantee that where key validation components for this zone appear in caches, they will include the

components for the successor key.

Not all rollovers go through the Preparation and Ready stages. The stages exist to facilitate rollover methods where a subset of Validation Components is introduced first and the final Validation Components are changed in an atomic manner. It is possible that a rollover goes through the Preparation and Ready stages multiple times.

Transition: The final Validation Components are added to the zone: The DNSKEY record is published in the zone, if it has not yet been introduced in previous Rollover Stages. In addition, a key that acts as ZSK is started to be used to sign RRsets, if it has not yet been started to do so in previous Rollover Stages. A key that acts as a KSK has its corresponding DS record submitted to the parent, if the DS record was not yet submitted in previous Rollover Stages. During this stage, it is not guaranteed that all RRs can be validated with the successor key information; some may only be able to be validated with information from the predecessor key. In some rollover scenario's, the Transition Stage is an atomic operation, where Validation Components of the successor key replace the Validation Components of its predecessor.

Transited: The Transited Stage always follows the Transition Stage. All the Validation Components of the successor key have been published long enough to guarantee that any cache that holds validation components for a RR in the zone will contain a copy of these components.

Revocation: If the key acts as a KSK, and it is known that the key is used as a [RFC5011] trust anchor, the predecessor key must be published for a period with the REVOKE bit set as a way of notifying validators that may have the key configured as a trust anchor, that is about to be removed from the zone.

Revoked: The Revoked Stage always follows the Revocation Stage. The revoked predecessor key has been published long enough to guarantee that RFC5011-aware validators have seen the key being revoked. Note that if the key is not used as an RFC5011 trust anchor, the rollover will not go through the Revocation and Revoked stages.

Withdrawal: At this point, there may still exist old Validation Components that belong to the predecessor key. Because a successor key is available, it is safe to withdraw all remaining old Validation Components.

Complete: All the Validation Components of the predecessor key have been removed from the zone long enough to guarantee that they have expired from caches.

3.2. ZSK Rollovers

The two most common rollover methods for ZSKs are Double-Signature and Pre-Publication. Both are described in RFC4641 [RFC4641]. [key-timing] also introduces ZSK Double-RRSIG rollover. These three rollover methods are shaped like this because different rollover considerations are being taken into account. Pre-Publication minimizes the number of signatures over the RRsets in the zone and DNS responses. Double-RRSIG keeps the size of the DNSKEY RRset to a minimum. Double-Signature is the fastest way to roll a ZSK, because no considerations are being taken into account.

3.2.1. Double-Signature

This involves introducing the new key into the zone and using it to create additional RRSIG records; the old key and existing RRSIG records are retained. During the period in which the zone is being signed, client validators are always able to validate RRSIGs: any combination of old and new DNSKEY RRset and RRSIG allows at least one signature to be validated.

Once the signing process is complete and enough time has elapsed to allow all old information to expire from caches, the old key and signatures can be removed from the zone. As before, during this period any combination of DNSKEY RRset and RRSIG will allow validation of at least one signature.

Double-Signature is the fastest way to rollover to a new key, since all new information is published right away. The drawback of this method is a noticeable increase in the size of the DNSSEC data, affecting both the overall size of the zone and the size of the responses.

The successor key Ks needs to be Known and Safe before Kc can be removed. First, all Validation Components of the successor key need to be introduced into the zone. Once all have been propagated, all information of Kc can be withdrawn from the zone.

The timeline diagram is shown below:

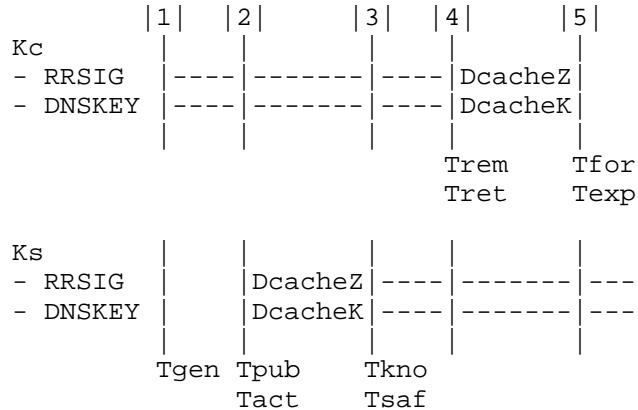


Figure: ZSK Double-Signature Rollover.

Generation Stage: Event 1

Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{RRSIG Hidden})$

$P(Ks) = \text{Uninformed}$

With the ZSK Double-Signature Rollover, all new Validation Components of the key Ks are going to be added to the zone and are allowed to propagate into the caches of validators. Thus, there is no need for Preparation and Ready Stages.

Transition Stage: Event 2

Key Ks is added to the DNSKEY RRset and is immediately used to sign the zone; existing signatures in the zone are maintained. This is Ks's publish time (Tpub) and Ks is said to be Published. It is also Ks's active time (Tact), the time when Ks is said to be Active. Because the Double-Signature rollover is in place, there are now temporarily two active keys.

$Tpub(Ks) \geq Tgen(Ks), Tact(Ks) == Tpub(Ks)$

$S(Ks) = (\text{DNSKEY Introduced}, \text{RRSIG Introduced})$

$P(Ks) = \text{Published Active}$

Transited Stage: Event 3

The information for Ks must be published long enough to ensure that the information have reached all validators that may have RRsets from

this zone cached. At the point in time that the DNSKEY RRset including Ks has been propagated and Ks is said to be Known (Tkno). At the point in time that the other RRsets including signatures of Ks have been propagated (Tsaf), Ks is said to be Safe.

$$\begin{aligned} \text{Tkno}(\text{Ks}) &\geq \text{Tpub}(\text{Ks}) + \text{DcacheK} \\ \text{Tsaf}(\text{Ks}) &\geq \text{Tact}(\text{Ks}) + \text{DcacheZ} \end{aligned}$$
$$\begin{aligned} \text{S}(\text{Ks}) &= (\text{DNSKEY Propagated}, \text{RRSIG Propagated}) \\ \text{P}(\text{Ks}) &= \text{Known Safe} \end{aligned}$$

Note that once the DNSKEY RRset containing both Kc and Ks has propagated to all caches, Kc can be retired (i.e. no longer used to sign RRsets). It does not matter if not all signatures of Ks have been Propagated, since the validator can validate RRsets with both Kc and Ks. If the validator fetches a RRset from the cache, it uses the DNSKEY of Kc for validation. Otherwise, it can use the DNSKEY of Ks.

Withdrawal Stage: Event 4

When the successor key Ks is said to be Propagated, Kc can be retired. And once there is a successor key that is said to be Safe, Kc can be removed. This is Kc's retire time (Tret) and Kc is said to be Retired. It is also Kc's removal time (Trem), the time that Kc is said to be Removed.

$$\begin{aligned} \text{Tret}(\text{Kc}) &\geq \text{Tkno}(\text{Ks}) \\ \text{Trem}(\text{Kc}) &\geq \text{MAX}(\text{Tkno}(\text{Ks}), \text{Tsaf}(\text{Ks})) \end{aligned}$$
$$\begin{aligned} \text{S}(\text{Kc}) &= (\text{DNSKEY Withdrawn}, \text{RRSIG Withdrawn}) \\ \text{P}(\text{Kc}) &= \text{Removed Retired} \end{aligned}$$

Complete Stage: Event 5

From the perspective of the authoritative server, the rollover is complete. After some delay, Kc and its signatures have expired from the caches. This delay is the maximum of DcacheZ, DcacheK. This is Tfor, the time that the key is said to be Forgotten and Texp, the time that the key is said to be Expired.

$$\begin{aligned} \text{Tfor}(\text{Kc}) &\geq \text{Trem}(\text{Kc}) + \text{DcacheK} \\ \text{Texp}(\text{Kc}) &\geq \text{Tret}(\text{Kc}) + \text{DcacheZ} \end{aligned}$$
$$\begin{aligned} \text{S}(\text{Kc}) &= (\text{DNSKEY Dead}, \text{RRSIG Dead}) \\ \text{P}(\text{Kc}) &= \text{Forgotten Expired} \end{aligned}$$

3.2.2. Pre-Publication

With Pre-Publication, the new key is introduced into the DNSKEY RRset, leaving the existing keys and signatures in place. This state

of affairs remains in place for long enough to ensure that any DNSKEY RRsets cached in client validators contain both keys. At that point signatures created with the old key can be replaced by those created with the new key, and the old signatures can be removed. During the re-signing process it doesn't matter which key an RRSIG record retrieved by a client was created with; clients with a cached copy of the DNSKEY RRset will have a copy containing both the old and new keys.

Once the zone contains only signatures created with the new key, there is an interval during which RRSIG records created with the old key expire from client caches. After this, there will be no signatures anywhere that were created using the old key, and it can be removed from the DNSKEY RRset.

Pre-Publication is more complex than Double-Signature - introduce the new key, approximately one TTL later sign the records, and approximately one TTL after that remove the old key. Although it takes more time than the Double-Signature method, it has the advantage that each RRset is signed with just one key. As a result, it has the advantage that the amount of DNSSEC data is kept to a minimum, reducing the impact on performance.

Only when Ks is said to be Known, Kc may be retired. Signatures may be retired all at once or may be incrementally replaced with signatures of Ks. However, during the transition all RRsets must either be signed with Kc or be signed with Ks. If Ks is considered to be Known and Safe, the DNSKEY record of Kc can be removed.

The timeline diagram looks like this:

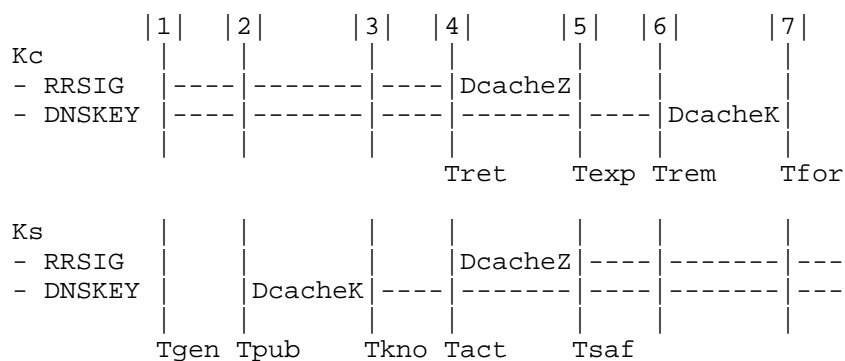


Figure: ZSK Pre-Publication Rollover.

With the ZSK Pre-Publication rollover, the DNSKEY record of Ks needs to be pre-published before the rollover can go into the Transition

Stage.

Generation Stage: Event 1

Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden)

P(Ks) = Uninformed

Preparation Stage: Event 2

The DNSKEY record of Ks is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the current KSK. The time at which this occurs is Ks's publication time (Tpub), and the key is now said to be Published. Note that the key is not yet used to sign records.

$T_{pub}(Ks) \geq T_{gen}(Ks)$

S(Ks) = (DNSKEY Introduced, RRSIG Hidden)

P(Ks) = Published

Ready Stage: Event 3

Before Ks can be used, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator that has a copy of the DNSKEY RRset in its cache also includes this key. In other words, that any prior cached information about the DNSKEY RRset has expired. After this delay, the key is said to be Known and could be used to sign records. The time at which this event occurs is Tkno, which is given by:

$T_{kno}(Ks) \geq T_{pub}(Ks) + D_{cacheK}$

S(Ks) = (DNSKEY Propagated, RRSIG Hidden)

P(Ks) = Known

At this point, the rollover is in the Ready Stage.

Transition Stage: Event 4

At some point in time, the decision is made to actually start signing the zone using the successor key. This will be when the current key has been in use for an interval equal to the ZSK lifetime. This point in time is Ks's active time (Tact), the time that Ks is said to be Active. It is also Kc's retire time (Tret), the time that Kc is said to be Retired.

$T_{act}(Ks) \geq T_{kno}(Ks), T_{ret}(Kc) == T_{act}(Ks)$

S(Kc) = (DNSKEY Propagated, RRSIG Withdrawn)
P(Kc) = Known Retired
S(Ks) = (DNSKEY Propagated, RRSIG Introduced)
P(Ks) = Known Active

Transited Stage: Event 5

Kc needs to be retained in the zone whilst any RRSIG records created by the retired key are still published in the zone or held in validator caches. In other words, Kc should be retained in the zone until all RRSIG records created by Ks have been propagated. This time is Ks's safe time (Tsaf), the time that Ks is considered to be Safe. Consequently, at the same time Kc is considered to be Expired.

$Tsaf(Ks) \geq Tact(Ks) + DcacheZ, Texp(Kc) == Tsaf(Ks)$

S(Kc) = (DNSKEY Propagated, RRSIG Dead)
P(Kc) = Known Expired
S(Ks) = (DNSKEY Propagated, RRSIG Propagated)
P(Ks) = Known Safe

Withdrawal Stage: Event 6

When all new signatures have been propagated, Kc can be removed from the zone and the DNSKEY RRset re-signed with the current KSK. This time is Kc's removal time (Trem), the time that Kc is considered to be Removed.

$Trem(Kc) \geq Tsaf(Ks)$

S(Kc) = (DNSKEY Withdrawn, RRSIG Dead)
P(Kc) = Removed Expired

Complete Stage: Event 7

From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record of Kc has expired from the caches. This is Tfor, and the key is said to be Forgotten.

$Tfor(Kc) \geq Trem(Kc) + DcacheK$

S(Kc) = (DNSKEY Dead, RRSIG Dead)
P(Kc) = Forgotten Expired

3.2.3. Double-RRSIG

[MM: Note it comes down to "signatures, generated with a key, whose public part is not published".]

This involves introducing the new signatures first, while existing signatures are being retained. This state of affairs remains in

place for long enough to ensure that all RRsets cached in client validators contain two signatures. The DNSKEY RR can now be switched. For the period of time before the predecessor key has been expired from all caches, it does not matter if the validator uses the cached key or the successor key that is in the zone. Both corresponding signatures can be retrieved from the cache or from the name server.

Once the signing process is complete and enough time has elapsed to allow all old information to expire from caches, the old signatures can be removed from the zone. As before, during this period any combination of DNSKEY RRset and RRSIG will allow validation of at least one signature.

Double-RRSIG is also more complex than Double-Signature - first introducing the signatures, then switch the key and finally remove the old signatures. It also takes more time than the Double-Signature method. The method will be used where it is desired not to publish the public data of both keys at the same time. As an advantage, the DNSKEY RRset is kept to a minimum which reduces the impact on priming performance. The disadvantage is that for a period, each RRset returned will be accompanied by two RRSIGs.

When Ks is said to be Safe, the DNSKEY record of Kc may be removed. At the same time that the DNSKEY record of Kc is removed, the DNSKEY record for Ks is introduced. If Ks is considered to be Known and Safe, Kc no longer needs to generate signatures.

The timeline diagram is shown below:

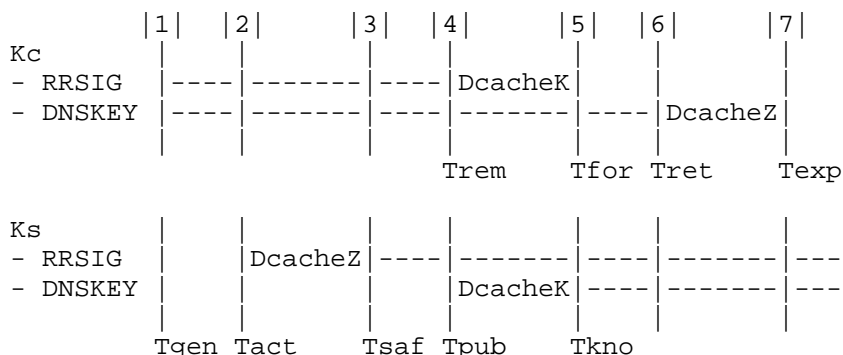


Figure: ZSK Double-RRSIG Rollover.

Generation Stage: Event 1

Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside

world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden)
P(Ks) = Uninformed

Preparation Stage: Event 2

The zone is signed with Ks but existing signatures are retained. The DNSKEY RR of Ks remains unpublished. The time at which this occurs is Ks's active time (Tact), and the key is now said to be Active.

Tact(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Hidden, RRSIG Introduced)
P(Ks) = Active

Ready Stage: Event 3

Before the DNSKEY from Kc can be switched to Ks, the signatures of Ks must be published for long enough (DcacheZ) to guarantee that any validator that has a copy of any RRset, also has both signatures. In other words, that any cached information is double signed. After this delay, the key is said to be Safe. The time at which this event occurs is Tsaf, which is given by:

Tsaf(Ks) >= Tact(Ks) + DcacheZ

S(Ks) = (DNSKEY Hidden, RRSIG Propagated)
P(Ks) = Safe

Transition Stage: Event 4

At some point in time, the decision is made to publish Ks. This point in time is Ks's publish time (Tpub), the time that Ks is said to be Published. At the same time, the DNSKEY RR of Kc is removed from the zone, and Kc is said to be Removed.

Tpub(Ks) >= Tsaf(Ks), Trem(Kc) == Tpub(Ks)

S(Kc) = (DNSKEY Removed, RRSIG Propagated)
P(Kc) = Removed Safe
S(Ks) = (DNSKEY Introduced, RRSIG Propagated)
P(Ks) = Published Safe

Transited Stage: Event 5

The signatures of Kc need to be retained in the zone until the DNSKEY RR has expired from all validator caches. When this happens, Ks is said to be Known (Tkno) and Kc is said to be Forgotten (Tfor).

Tfor(Kc) >= Trem(Kc) + DcacheK
Tkno(Ks) >= Tpub(Ks) + DcacheK

S(Kc) = (DNSKEY Dead, RRSIG Propagated)
P(Kc) = Forgotten Safe
S(Ks) = (DNSKEY Propagated, RRSIG Propagated)
P(Ks) = Known Safe

Withdrawal Stage: Event 6

The signatures of Kc can be removed when the DNSKEY RR of Ks has been propagated. This time is Kc's retire time (Tret), the time that Kc is considered to be Retired.

$Tret(Kc) \geq Tsaf(Ks)$

S(Kc) = (DNSKEY Dead, RRSIG Withdrawn)
P(Kc) = Forgotten Retired

Complete Stage: Event 7

From the perspective of the authoritative server, the rollover is complete. After some delay, all signatures of Kc have expired from the caches. This is Texp, and the key is said to be Expired.

$Texp(Kc) \geq Tret(Kc) + DcacheZ$

S(Kc) = (DNSKEY Dead, RRSIG Dead)
P(Kc) = Forgotten Expired

3.3. KSK Rollovers

The most common rollover method for KSKs is Double-Signature, described in RFC4641 [RFC4641]. Two more methods are identified in [key-timing]: Double-DS and Double-RRset. Double-RRset is the fastest way to rollover a KSK, while Double-Signature minimizes the number of required interactions to the parent, and Double-DS keeps the DNSKEY RRset as small as possible.

Note that with the KSK rollovers, it is out of scope whether the information within the zone is authentic. It is assumed that there exists one or more ZSKs in the DNSKEY RRset that takes care of this during the rollover.

3.3.1. Double-RRset

With Double-RRset, the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key, and the new DS record added to the parent zone. After waiting a suitable interval for the old DS and DNSKEY RRsets to expire from validator caches, the old DNSKEY and DS record are removed.

Ks needs to be Known and SafeDS, before Kc can be removed. First,

all new information for Ks need to be introduced into the zone. Once all have been propagated, all information of Kc can be withdrawn from the zone.

The timeline diagram looks like this:

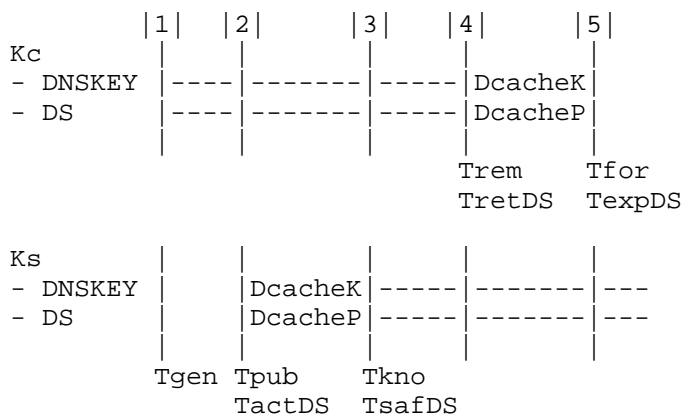


Figure: KSK Double-RRset Rollover.

Generation Stage: Event 1

Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Uninformed}$

With the KSK Double-RRset Rollover, all new Validation Components of the key Ks are going to be added to the zone and are allowed to propagate into the caches of validators. Thus, there is no need for a Preparation Stage.

Transition Stage: Event 2

Ks is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by all currently active KSKs (including Kc and Ks). In addition, the DS record is submitted to the parent. This is Ks's publish time (Tpub), the time that Ks is said to be Published. It is also Ks's submit time (TactDS), the time that the DS record of Ks is Submitted (ActiveDS).

$Tpub(Ks) \geq Tgen(Ks), TactDS(Ks) == Tpub(Ks)$

$S(Ks) = (\text{DNSKEY Introduced}, \text{DS Introduced})$

$P(Ks) = \text{Published ActiveDS}$

After the registration delay, the DS is published in the parent.

Transited Stage: Event 3

The information for Ks must be published long enough to ensure that the information have reached all validators that may have the DNSKEY or DS RRset from this zone cached. At the point in time that the DNSKEY RRset including Ks has been propagated (Tkno), Ks is said to be Known. At the point in time that the DS RRset of Ks has been propagated (Tsaf), Ks is said to be SafeDS.

$$Tkno(Ks) \geq Tpub(Ks) + DcacheK, TsafDS(Ks) \geq TactDS(Ks) + DcacheP$$
$$S(Ks) = (DNSKEY \text{ Propagated}, DS \text{ Propagated})$$
$$P(Ks) = \text{Known SafeDS}$$

Note that the request to the parent to withdraw the DS record of Kc can already be made after DcacheK. It does not matter if the DS record for Ks has not yet been propagated, since the validator can authenticate the DNSKEY RRset with both Kc and Ks. If the validator fetches a DS RRset from the cache, it uses Kc. Otherwise, it can use Ks.

Withdrawal Stage: Event 4

Once the successor key Ks is said to be Known, the DS record of Kc can be withdrawn. If Ks is also said to be SafeDS, the DNSKEY record of Kc can be removed from the zone. This is Kc's retire time (Tret), the time that Kc is said to be RetiredDS. It is also Kc's removal time (Trem), the time that Kc is said to be Removed.

$$TretDS(Kc) \geq Tkno(Ks)$$
$$Trem(Kc) \geq \text{MAX}(TsafDS(Ks), Tkno(Ks))$$
$$S(Kc) = (DNSKEY \text{ Withdrawn}, DS \text{ Withdrawn})$$
$$P(Kc) = \text{Removed RetiredDS}$$

Complete Stage: Event 5

From the perspective of the authoritative server, the rollover is complete. After some delay, Kc and its DS have also expired from the caches.

$$Tfor(Kc) \geq Trem(Kc) + DcachK$$
$$TexpDS(Kc) \geq TretDS(Kc) + DcacheP$$
$$S(Kc) = (DNSKEY \text{ Dead}, DS \text{ Dead})$$
$$P(Kc) = \text{Forgotten Expired}$$

3.3.2. Double-Signature

With Double-Signature, the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key. After waiting for the old RRset to expire from caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset.

If the number of interactions to the parent needs to be minimized, this rollover method is preferred over the Double-RRset method. As a consequence, the DS record of Ks can be submitted to the parent only if it is safe to withdraw the DS record of Kc.

When Ks is said to be Known, the DS record can be changed. When Ks is considered to be Known and SafeDS, the DNSKEY record of Kc can be removed.

The timing diagram for such a rollover is:

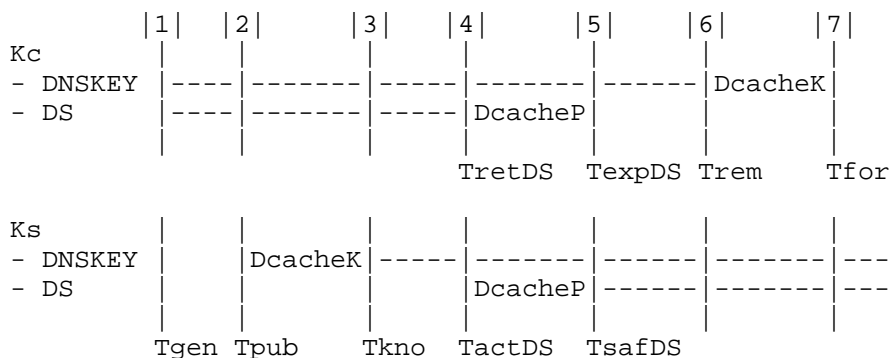


Figure: KSK Double-Signature Rollover.

Generation Stage: Event 1

Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Uninformed}$

Preparation Stage: Event 2

Ks is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by Ks and all currently active KSKs (including Kc). This is the publication time (Tpub), the time that Ks is said to be Published.

$T_{\text{pub}}(Ks) \geq T_{\text{gen}}(Ks)$

S(Ks) = (DNSKEY Introduced, DS Hidden)
P(Ks) = Published

Ready Stage: Event 3

Before the corresponding DS can be submitted, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator that has a copy of the DNSKEY RRset also includes this key. In other words, that any prior cached information about the DNSKEY RRset has expired. This time is Tkno and Ks is said to be Known.

$Tkno(Ks) \geq Tpub(Ks) + DcacheK$

S(Ks) = (DNSKEY Propagated, DS Hidden)
P(Ks) = Known

Transition Stage: Event 4

At some later time, the DS RR corresponding to Ks is submitted to the parent zone for publication, with a request that it replaces the DS RR corresponding to Kc. This time is Ks's submit time (TactDS), the time that Ks is considered to be Submitted. It is also Kc's retire time (TretDS), the time that Kc is considered to be RetiredDS.

$TactDS(Ks) \geq Tkno(Ks)$
 $TretDS(Kc) == TactDS(Kc)$

S(Kc) = (DNSKEY Propagated, DS Withdrawn)
P(Ks) = Known RetiredDS
S(Ks) = (DNSKEY Propagated, DS Introduced)
P(Ks) = Known ActiveDS

After the registration delay, the DS is published in the parent.

Transited Stage: Event 5

All validators that have the DS RRset cached will have a copy that includes the new DS record. This is Ks's safe time (TsafDS), the time that the new KSK is said to be SafeDS. Consequently, Kc is said to be ExpiredDS (TexpDS).

$TsafDS(Ks) \geq TactDS(Ks) + DcacheP$
 $TexpDS(Kc) \geq TretDS(Kc) + DcacheP$

S(Kc) = (DNSKEY Propagated, DS Dead)
P(Kc) = Known ExpiredDS
S(Ks) = (DNSKEY Propagated, DS Propagated)
P(Ks) = Known SafeDS

Withdrawal Stage: Event 6

When the new DS record has been propagated, the DNSKEY record of Kc

can be removed from the zone. This is Kc's removal time (Trem), the time that Kc is said to be Removed.

$Trem(Kc) \geq TsafDS(Ks)$

$S(Kc) = (DNSKEY\ Withdrawn, DS\ Dead)$

$P(Kc) = Removed\ ExpiredDS$

Complete Stage: Event 7

From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record of Kc has also expired from the caches.

$Tfor(Kc) \geq Trem(Kc) + DcacheK$

$S(Kc) = (DNSKEY\ Dead, DS\ Dead)$

$P(Kc) = Forgotten\ ExpiredDS$

3.3.3. Double-DS

In this case, first the new DS record is published. After waiting for this change to propagate into the caches of all validators, the KSK is changed. After waiting another interval, during which the old DNSKEY RRset expires from caches, the old DS record is removed.

If the size of the DNSKEY RRset needs to be minimized, this rollover method is preferred over Double-RRset. It does require the additional administrative overhead of two interactions with the parent to roll a KSK.

When Ks is said to be SafeDS, the DNSKEY record of Kc can be removed. At the same time, the DNSKEY record of Ks can be introduced. When Ks is considered to be Known and SafeDS, the DS record of Kc can be removed.

The timeline diagram looks like this:

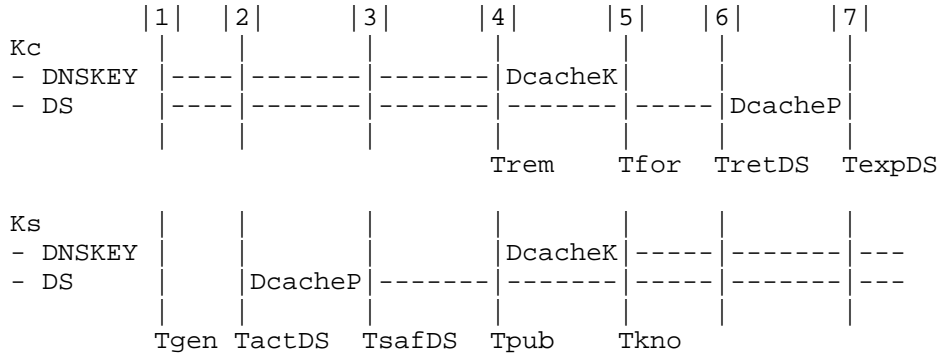


Figure: KSK Double-DS Rollover.

Generation Stage: Event 1

Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Uninformed}$

Preparation Stage: Event 2

Before the new key Ks can be introduced into the zone, the new DS record needs to be submitted. This is allowed, because there exists a valid chain of trust for the same algorithm (with the current key Kc). This is Ks's submit time (TactDS), the time that the DS record of Ks was submitted and is said to be ActiveDS.

$TactDS(Ks) \geq Tgen(Ks)$

$S(Ks) = (\text{DNSKEY Hidden}, \text{DS Introduced})$

$P(Kc) = \text{ActiveDS}$

After some delay, the DS becomes available in the parent zone.

Ready Stage: Event 3

The new DS RRset has been propagated. This is Ks's safe time (TsafDS), the time that Ks is said to be SafeDS.

$TsafDS(Ks) \geq TactDS(Ks) + DcacheP$

$S(Ks) = (\text{DNSKEY Hidden}, \text{DS Propagated})$

$P(Ks) = \text{SafeDS}$

Transition Stage: Event 4

Because there are now two trust anchors a validator can use, the DNSKEY record of Kc can be switched with the DNSKEY record of Ks. Kc becomes retired and the DNSKEY RRset is only signed with Ks. This time is Ks's publish time (Tpub), the time that Ks is said to be Published. It is also Kc's removal time (Trem), the time that Kc is said to be Removed.

Tpub(Ks) >= TsafDS(Ks)
Trem(Kc) == Tpub(Ks)

S(Kc) = (DNSKEY Withdrawn, DS Propagated)
P(Kc) = Removed SafeDS
S(Ks) = (DNSKEY Introduced, DS Propagated)
P(Ks) = Published SafeDS

Transited Stage: Event 5

Before the DS record of Kc can be withdrawn, Kc will have to expire from validator caches. When the DNSKEY RRset that includes Kc has expired from caches, Kc is said to be Forgotten and Ks is said to be Known. This happens at Ks's known time, given by:

Tkno(Ks) >= Tpub(Ks) + DcacheK
Tfor(Kc) == Tkno(Ks)

S(Kc) = (DNSKEY Dead, DS Propagated)
P(Kc) = Forgotten SafeDS
S(Ks) = (DNSKEY Propagated, DS Propagated)
P(Ks) = Known SafeDS

Withdrawal Stage: Event 6

Now that there is a key Ks that is said to be Propagated and SafeDS, the DS record of Kc can be withdrawn. This is Kc's retire time (TretDS), the time that there is no need for a secure delegation for Kc anymore.

TretDS(Kc) >= Tkno(Ks)

S(Kc) = (DNSKEY Dead, DS Withdrawn)
P(Kc) = Forgotten RetiredDS

Complete Stage: Event 7

From the perspective of the authoritative server, the rollover is complete. After some delay, The DS record of Kc has expired from the caches. This is Texp, given by:

Texp(Kc) >= Tret(Kc) + DcacheP

S(Kc) = (DNSKEY Dead, DS Dead)

$P(Kc) = \text{Forgotten ExpiredDS}$

3.3.4. Interaction with Configured Trust Anchors

Zone managers may want to take into account the possibility that some validators may have their KSK configured as a trust anchor directly, as described in RFC5011 [RFC5011]. This influences the value of DcacheK, the time to guarantee that any validator that has a copy of the newest DNSKEY RRset.

3.3.4.1. Adding a KSK

When the new key is introduced, the delay DcacheK between Tpub and Tkno is also subject to the property:

$$DcacheK' = \text{MAX}(DcacheK, 2 * (\text{queryInterval} + x * \text{retryTime}) + c)$$

The right hand side of this expression is two times the Active Refresh time defined in section 2.3 in RFC5011 [RFC5011]. This ensures that the successor key is at least seen twice by RFC5011-aware validators. The parameter x is the maximum number of retries that is taken as a safety margin, in case an Active Refresh fails. The parameter c is a constant that can be taken as an additional safety margin.

Most probably, this delays the time when a key is said to be Known.

3.3.4.2. Removing a KSK

When the current key is ready to be removed from the zone, it is instead going to be revoked. The REVOKE bit is set and the key is published for DcacheK' time:

$$DcacheK' = \text{MAX}(DcacheK, (\text{queryInterval} + x * \text{retryTime}) + c)$$

The right hand side of this expression is the Active Refresh time defined in section 2.3 in RFC5011 [RFC5011]. This ensures that the revoked key is at least seen once by RFC5011-aware validators.

After that delay, every RFC5011-aware validator has seen the revoked key and the DNSKEY record may be removed from the zone. Another DcacheK delay, the key has fully expired from all the validator caches.

3.4. Rollovers in a Single Type Signing Scheme

Previous sections described the possible ways to roll keys that have one key type (either ZSK or KSK). In situations where a Single Type

Signing Scheme (STSS) is used, one key is used both as ZSK and KSK. This key is responsible for authenticating information within the zone, as well as authenticating the DNSKEY RRset. STSS Rollovers can be constructed by combining a ZSK rollover method with a KSK rollover method. However, not all combinations are possible. For example, the ZSK Double-RRSIG rollover is only suitable for combining with the KSK Double-DS rollover, because both keep the DNSKEY RRset to a minimum size. The other rollovers are ruled out because they require a period where both the DNSKEY record of the current key and its successor are being served at the same time.

The ZSK Pre-Publication method is suitable for combining with the KSK Double-RRset and KSK Double-Signature rollover methods, but does not gain any advantages when combined with the KSK Double-RRset. In both cases the DNSKEY record needs to be post-published, taking a similar amount of time. However, the KSK Double-RRset requires two interactions with the parent, while the KSK Double-Signature only involves one interaction. Therefore, the ZSK Pre-Publication rollover combined with the KSK Double-RRset is left out of this document.

The ZSK Double-Signature method is suitable for combining with both the KSK Double-RRset and the KSK Double-Signature method.

To conclude, there are four different STSS rollover methods.

3.4.1. Double-RRset

This is a combination of the ZSK Double-Signature rollover and the KSK Double-RRset rollover. The new key is added to the DNSKEY RRset, and all RRsets - including the DNSKEY RRset - are then signed with both the old and new key, and the new DS record added to the parent zone. After waiting a suitable interval for the old DS and all zone RRsets to expire from validator caches, the old DNSKEY and DS record are removed.

Double-RRset is the fastest way to replace keys in a Single Type Signing Scheme. However, the disadvantages are that it requires two signatures and two keys during the period of the rollover, as well as two interactions with the parent.

The timeline diagram looks like this:

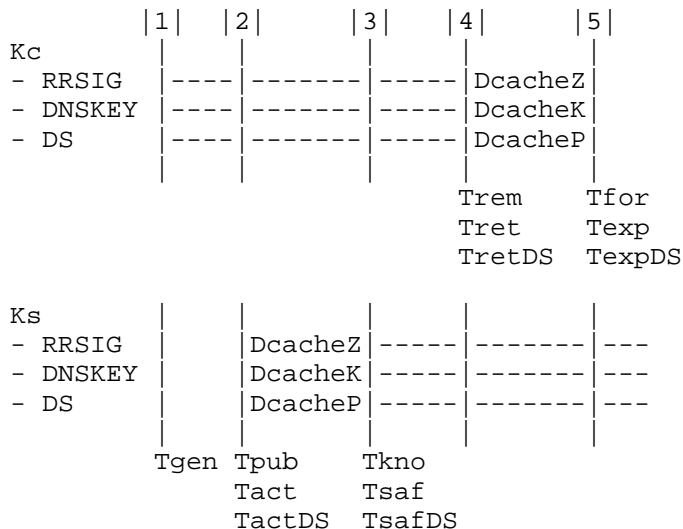


Figure: STSS Double-RRset Rollover.

The rollover method is almost the same as that of the KSK Double-RRset rollover, except now also DcacheZ has to be taken into account.

3.4.2. Double-Signature

This is a combination of the ZSK Double-Signature rollover and the KSK Double-Signature rollover. The new key is added to the DNSKEY RRset and all RRsets are then signed with both the old and new key. After waiting for the old RRsets to expire from caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the DNSKEY RRset, and all RRsets are signed with the new key only.

This rollover minimizes the number of interactions with the parent zone. However, for the period of the rollover all RRsets are still signed with two keys, so increasing the size of the zone and the size of the response.

The timing diagram for such a rollover is:

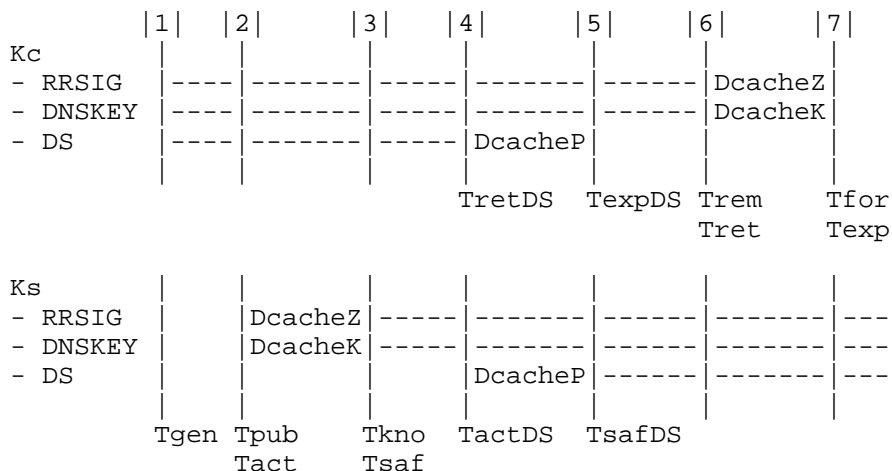


Figure: STSS Double-Signature Rollover.

The rollover method is almost the same as that of the KSK Double-RRset rollover, except now also DcacheZ has to be taken into account.

3.4.3. Pre-Publication

This is a combination of the ZSK Pre-Publication rollover and the KSK Double-Signature rollover and requires only one interaction with the parent. In addition, the non-DNSKEY RRsets require only one signature during the rollover. If speed is not an issue, this rollover method might be the way to go in a STSS environment, since it optimizes in both size and interactions with the parent.

The new key is added to the DNSKEY RRset and the DNSKEY RRset is then signed with both the old and new key. Other RRsets will only be signed with the old key. Only after the DS has been switched, the signatures of other RRsets are replaced with that of the new key. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset, and is signed with the new key only.

The timeline diagram looks like this:

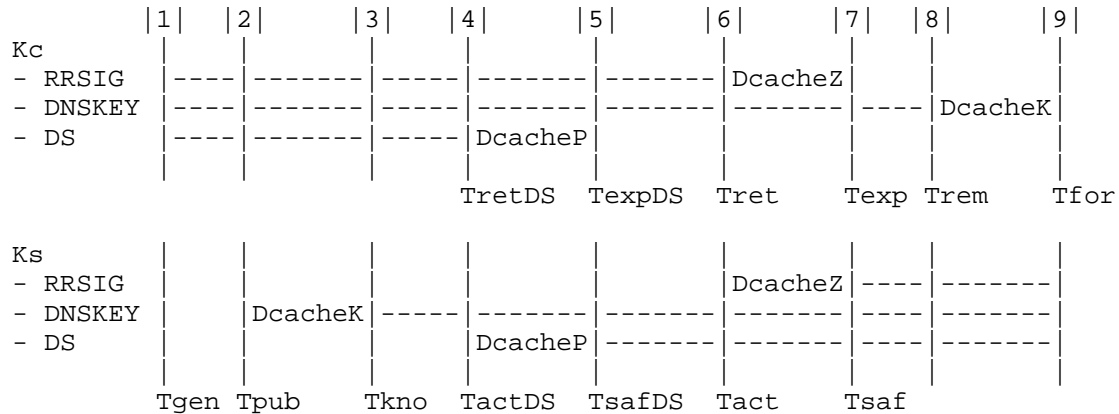


Figure: STSS Pre-Publication Rollover.

Generation Stage: Event 1

Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{RRSIG Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Uninformed}$

Preparation Stage: Event 2

The DNSKEY record of Ks is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the Ks and all other current KSKs (including Kc). The time at which this occurs is Ks's publication time (Tpub), and the key is now said to be Published. Note that the key is not yet used to sign other RRsets.

$T_{pub}(Ks) \geq T_{gen}(Ks)$

$S(Ks) = (\text{DNSKEY Introduced}, \text{RRSIG Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Published}$

Ready Stage: Event 3

Before the DS record may be switched, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator that has a copy of the DNSKEY RRset also includes this key. After this delay, the key is said to be Known and the DS record may be submitted. The time at which this event occurs is Ks's known time (Tkno), which is given by:

$T_{kno}(Ks) \geq T_{pub}(Ks) + D_{cacheK}$

S(Ks) = (DNSKEY Propagated, RRSIG Hidden, DS Hidden)
P(Ks) = Known

Preparation Stage (2): Event 4

At Ks's submit time (TactDS), the DS RR corresponding to Ks is submitted to the parent zone for publication, with a request that it replaces the DS RR corresponding to Kc. Ks is said to be ActiveDS. This time is also Kc's retire time (TretDS) and Kc is said to be RetiredDS.

TactDS(Ks) >= Tkno(Ks) TretDS(Kc) == TactDS(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Propagated, DS Withdrawn)
P(Kc) = Known Safe RetiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Hidden, DS Introduced)
P(Ks) = Known ActiveDS

Some time later, the new DS RRset is published at the parent.

Ready Stage (2): Event 5

All validators use the DS RRset that includes a copy of the DS record of Ks. At this time, Ks's safe time (TsafDS), Ks is said to be SafeDS. But Kc is still used as the ZSK.

TsafDS(Ks) >= TactDS(Ks) + DcacheP
TexpDS(Kc) >= TretDS(Kc) + DcacheP

S(Kc) = (DNSKEY Propagated, RRSIG Propagated, DS Dead)
P(Kc) = Known Safe ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Hidden, DS Propagated)
P(Ks) = Known SafeDS

Transition Stage: Event 6

At some point in time, the decision is made to actually start signing the zone using the successor key. This will be when the current key has been in use for an interval equal to the key lifetime. This point in time is Ks's active time (Tact), the time that Ks is said to be Active. It is also Kc's retire time (Tret), the time that Kc is said to be Retired.

Tact(Ks) >= TsafDS(Ks)
Tret(Kc) == Tact(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Withdrawn, DS Dead))
P(Kc) = Known Retired ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Introduced, DS Propagated))
P(Ks) = Known Active SafeDS

Transited Stage: Event 7

Kc needs to be retained in the zone whilst any RRSIG records created by the retired key are still published in the zone or held in validator caches. In other words, Kc should be retained in the zone until all RRSIG records created by Ks have been propagated. This time is Ks's safe time (Tsaf), the time that Ks is considered to be Safe, and Kc's expiration time (Texp), the time that Kc is considered to be Expired.

$Tsaf(Ks) \geq Tact(Ks) + DcacheZ$
 $Texp(Kc) == Tsaf(Ks)$

$S(Kc) = (DNSKEY \text{ Propagated}, RRSIG \text{ Dead}, DS \text{ Dead})$
 $P(Kc) = \text{Known Expired ExpiredDS}$
 $S(Ks) = (DNSKEY \text{ Propagated}, RRSIG \text{ Propagated}, DS \text{ Propagated})$
 $P(Ks) = \text{Known Safe SafeDS}$

Withdrawal Stage: Event 8

When all new signatures have been propagated, Kc can be removed from the zone and the DNSKEY RRset re-signed with the current KSK. This time is Kc's removal time (Trem), the time that Kc is considered to be Removed.

$Trem(Kc) \geq Tsaf(Ks)$

$S(Kc) = (DNSKEY \text{ Withdrawn}, RRSIG \text{ Dead}, DS \text{ Dead})$
 $P(Kc) = \text{Removed Expired ExpiredDS}$

Complete Stage: Event 9

From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record of Kc has expired from the caches. This is Tfor, the time that the key is said to be Forgotten.

$Tfor(Kc) \geq Trem(Kc) + DcacheK$

$S(Kc) = (DNSKEY \text{ Dead}, RRSIG \text{ Dead}, DS \text{ Dead})$
 $P(Kc) = \text{Forgotten Expired ExpiredDS}$

3.4.4. Double-DS

This is a combination of the ZSK Double-RRSIG rollover and the KSK Double-DS rollover. This keeps the DNSKEY RRset to a minimum size, but at the cost of double signatures in the zone and the necessity of two interactions with the parent.

The new signatures are added to the zone and the new DS is submitted. Once all signatures and the DS record have been propagated, the

DNSKEY is switched. After waiting a further interval for this switch to be reflected in caches, the old signatures are removed and the old DS record is withdrawn from the parent zone.

The timeline diagram looks like this:

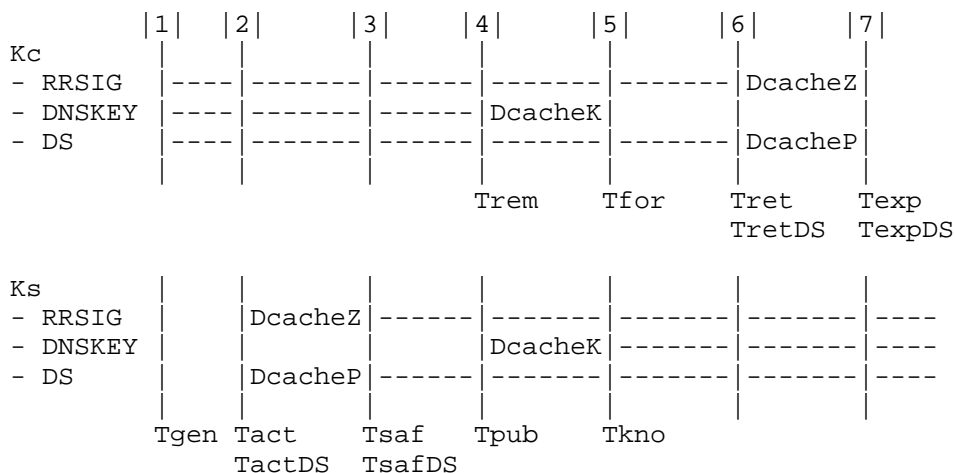


Figure: STSS Double-DS Rollover.

Generation Stage: Event 1

Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

$S(Ks) = (\text{DNSKEY Hidden}, \text{RRSIG Hidden}, \text{DS Hidden})$

$P(Ks) = \text{Uninformed}$

Preparation Stage: Event 2

Before the new key Ks can be introduced into the zone, the new signatures must be introduced and the new DS record must be submitted. This time is Ks's active time (Tact), the time that Ks is said to be Active. It is also Ks's submit time (TactDS), the time that the DS record of Ks was submitted and is said to be ActiveDS.

$Tact(Ks) \geq Tgen(Ks)$

$TactDS(Ks) \geq Tgen(Ks)$

$S(Ks) = (\text{DNSKEY Hidden}, \text{RRSIG Introduced}, \text{DS Introduced})$

$P(Kc) = \text{Active ActiveDS}$

After some delay, the DS becomes available in the parent zone.

Ready Stage: Event 3

The new signatures and the new DS RRset have been propagated. This is Ks's safe time (Tsaf, TsafDS), the time that Ks is said to be Safe and SafeDS.

$$Tsaf(Ks) \geq Tact(Ks) + DcacheZ$$
$$TsafDS(Ks) \geq TactDS(Ks) + DcacheP$$
$$S(Ks) = (\text{DNSKEY Hidden, RRSIG Propagated, DS Propagated})$$
$$P(Ks) = \text{Safe SafeDS}$$

Transition Stage: Event 4

Because there are now two trust anchors a validator can use, the DNSKEY record of Kc can be switched with the DNSKEY record of Ks. This time is Ks's publish time (Tpub), the time that Ks is said to be Published. It is also Kc's removal time (Trem), the time that Kc is removed from the zone.

$$Tpub(Ks) \geq \text{MAX}(TsafDS(Ks), Tsaf(Ks))$$
$$Trem(Kc) == Tpub(Ks)$$
$$S(Kc) = (\text{DNSKEY Withdrawn, RRSIG Propagated, DS Propagated})$$
$$P(Kc) = \text{Removed Safe SafeDS}$$
$$S(Ks) = (\text{DNSKEY Introduced, RRSIG Propagated, DS Propagated})$$
$$P(Ks) = \text{Published Safe SafeDS}$$

Transited Stage: Event 5

When the signatures of Kc and its corresponding DS record have expired from the caches, the DNSKEY record of Kc can be withdrawn from the zone. When the DNSKEY RRset that includes Kc has been expired, Ks is said to be Known and Kc is said to be Removed. This happens at Ks's known time, given by:

$$Tkno(Ks) \geq Tpub(Ks) + DcacheK, Trem(Kc) == Tkno(Ks)$$
$$S(Kc) = (\text{DNSKEY Dead, RRSIG Propagated, DS Propagated})$$
$$P(Kc) = \text{Forgotten Safe SafeDS}$$
$$S(Ks) = (\text{DNSKEY Propagated, RRSIG Propagated, DS Propagated})$$
$$P(Ks) = \text{Known Safe SafeDS}$$

Withdrawal Stage: Event 6

Ks is said to be Propagated and SafeDS, and the signatures and DS record of Kc may be withdrawn. This is this Kc's retire time (Tret, TretDS), the time Kc is said to be Retired and RetiredDS.

$$Tret(Kc) \geq Tkno(Ks)$$
$$TretDS(Kc) \geq Tkno(Ks)$$

$S(Kc) = (\text{DNSKEY Dead}, \text{RRSIG Withdrawn}, \text{DS Withdrawn})$
 $P(Kc) = \text{Forgotten Retired RetiredDS}$

Complete Stage: Event 7

From the perspective of the authoritative server, the rollover is complete. After some delay, The signatures of Kc and its corresponding DS record have expired from the caches.

$\text{Texp}(Kc) \geq \text{Tret}(Kc) + \text{DcacheZ}$
 $\text{TexpDS}(Kc) \geq \text{TretDS}(Kc) + \text{DcacheP}$

$S(Kc) = (\text{DNSKEY Dead}, \text{RRSIG Dead}, \text{DS Dead})$
 $P(Kc) = \text{Forgotten Expired ExpiredDS}$

3.5. Stand-by Keys

Although keys will usually be rolled according to some regular schedule, there may be occasions where an emergency rollover is required, e.g. if the active key is suspected of being compromised. The aim of the emergency rollover is to allow the zone to be re-signed with a new key as soon as possible. As a key must be ready to sign the zone, having at least one additional key (a stand-by key) in this state at all times will minimise delay.

In the case of a ZSK, a stand-by key only makes sense with the Pre-Publication method, since with the Double-Signature and Double-RRSIG methods, the stand-by key would be used for signing. The goal is to make the stand-by key Known. This goal is reached at Tkno, step 3 in the Pre-Publication method timeline diagram.

A successor key must always be published soon enough so that the key lifetime of the predecessor key does not expire. As a consequence, a stand-by ZSK Ks must at latest be published DcacheK delay before the lifetime of the predecessor ZSK Kc has reached:

$\text{Tpub}(Ks) \leq \text{Tact}(Kc) + \text{Lzsk} - \text{DcacheK}$

Here, Lzsk is the lifetime of ZSKs according to policy.

In the case of a KSK, a stand-by key only makes sense with the Double-DS method, since in the other cases, the key would be needed to sign the DNSKEY RRset. The goal is to get the stand-by key in the SafeDS state. This goal is reached at TsafDS, step 3 in the Double-DS method timeline diagram.

The DS record for the stand-by KSK Ks should be propagated to the caches before the key lifetime of the predecessor KSK Kc expires:

$$\text{TactDS}(K_s) \leq \text{Tact}(K_c) + \text{Lksk} - \text{DcacheP}$$

Here, Lksk is the lifetime of KSKs according to policy.

Because a stand-by KSK only makes sense with the Double-DS method, stand-by keys in a STSS is not applicable. This is because the Double-DS method is not easy integratable with one of the ZSK rollover methods.

4. Policy rollover

Besides (un)scheduled key rollovers, changes in policy may occur. The initial transition is enabling DNSSEC. The counterpart, disabling DNSSEC, is also possible. Two other examples of policy changes are algorithm rollover and changing signing schemes.

4.1. Enabling DNSSEC

When a zone makes the transition from going insecure to secure, the initial set of keys safely need to be introduced into the zone. The goals of this event is to make a ZSK (Kz) and a KSK (Kk) both Known and Safe.

A zone must be fully signed, before the DS associated with the initial KSK is published. The ZSK and KSK can be the same key, for example in a Single Type Signing Scheme.

The timeline diagram is shown below:

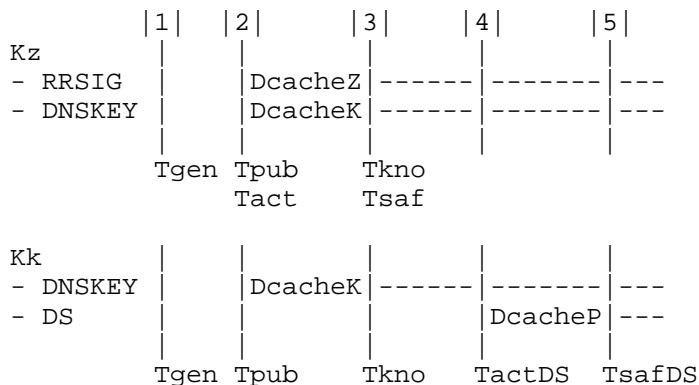


Figure: Enabling DNSSEC.

Generation Stage: Event 1

Kk and Kz are generated. The time when this happens is Tgen, the time that the keys were generated (note that Tgen for Kk could be

different that Tgen for Kz).

```
S(Kk) = (DNSKEY Hidden, DS Hidden)
P(Kk) = Uninformed
S(Kz) = (DNSKEY Hidden, RRSIG Hidden)
P(Kk) = Uninformed
```

Preparation Stage: Event 2

The keys are put into the zone and are immediately used for signing. Because there exists no pointer to the fact that our zone is DNSSEC enabled, the DNSKEY and RRSIG records may be introduced at the same time. This is the publish time (Tpub), the time that the keys are Published. It is also Kz' active time (Tact), the time that Kz is said to be Active.

```
Tpub(Kk) >= Tgen(Kk)
Tpub(Kz) >= Tgen(Kz)
Tact(Kz) == Tpub(Kz)
```

```
S(Kk) = (DNSKEY Introduced, DS Hidden)
P(Kk) = Published
S(Kz) = (DNSKEY Introduced, RRSIG Introduced)
P(Kz) = Published Active
```

Ready Stage: Event 3

Before the DS record can be committed, Kz must be considered Known and Safe. This time is Kz' known time (Tkno).

```
Tkno(Kk) >= Tpub(Kk) + DcacheP
Tkno(Kk) == Tkno(Kz)
Tsaf(Kz) >= Tact(Kz) + DcacheZ
```

```
S(Kk) = (DNSKEY Propagated, DS Hidden)
P(Kk) = Known
S(Kz) = (DNSKEY Propagated, RRSIG Propagated)
P(Kz) = Known Safe
```

Because this is the first DNSKEY for this zone, the Dttl for the DNSKEY RRset is Ingc, the negative cache interval from the zone's SOA record, calculated according to RFC2308 [RFC2308] as the minimum of the TTL of the SOA record itself and the MINIMUM field in the record's parameters:

```
Ingc = min(TTL(SOA), MINIMUM)
```

Transition Stage: Event 4

The DNSKEY RRset and all RRSIG records have reached the caches, and the DS record can be submitted to the parent. This is TactDS, the

time that the DS has been submitted to the parent.

$TactDS(Kk) \geq Tkno(Kk)$

$S(Kk) = (\text{DNSKEY Propagated}, \text{DS Introduced})$

$P(Kk) = \text{Known ActiveDS}$

Transited Stage: Event 5

The DS has been published in the parent zone. Some more time later, all validators that have a copy of the DS RRset have one that includes the DS record of Kk.

$TsafDS(Kk) \geq TactDS(Kk) + DcacheP$

$S(Kk) = (\text{DNSKEY Propagated}, \text{DS Propagated})$

$P(Kk) = \text{Known SafeDS}$

Because this is the first DS for this zone, the Dttl for the DS RRset is Ingc, for the same reason as in step 3 for the DNSKEY RRset.

4.2. Disabling DNSSEC

When a zone decides for whatever reason to go back to the Insecure status, the set of keys safely need to be removed from the zone. It is assumed that there is a KSK (Kk) and a ZSK (Kz) that are Known and Safe. The goals of this event are to make Kk and Kz both Forgotten and Expired.

The timeline diagram is shown below:

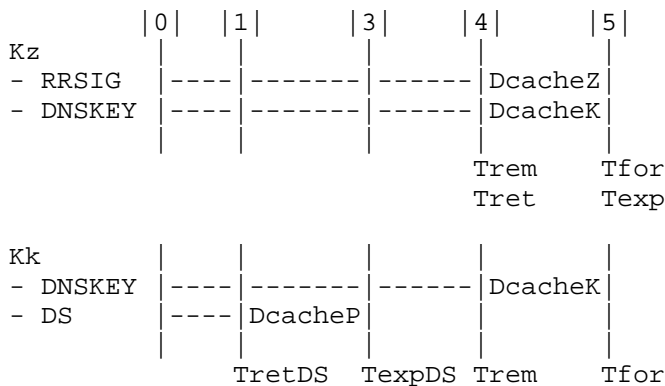


Figure: Disabling DNSSEC.

Transition Stage: Event 1

The DS record of Kk needs to be withdrawn. This time is Kk' retire time (TretDS), the time that Kk is said to be RetiredDS.

S(Kk) = (DNSKEY Propagated, DS Withdrawn)
P(Kk) = Known RetiredDS

Transited Stage: Event 2

First, the DS record of Kk must expire from all validator caches. This time is Kk' expire time (TexpDS), the time that Kk is said to be ExpiredDS.

$TexpDS(Kk) \geq TretDS(Kk) + DcacheP$

S(Kk) = (DNSKEY Propagated, DS Dead)
P(Kk) = Known ExpiredDS

Withdrawal Stage: Event 3

When no secure chain of trust to Kk exists anymore, the DNSKEY records of both keys and all RRSIG records can be removed from the zone. This time is Trem, the time that the keys are removed from the zone.

$Trem(Kk) \geq TexpDS(Kk)$
 $Trem(Kz) == Trem(Kk)$
 $Tret(Kz) == Trem(kz)$

S(Kk) = (DNSKEY Withdrawn, DS Dead)
P(Kk) = Removed ExpiredDS
S(Kz) = (DNSKEY Withdrawn, RRSIG Withdrawn)
P(Kz) = Removed Retired

Complete Stage: Event 4

After some delay, all information about the keys have expired from the caches.

$Tfor(Kk) \geq Trem(Kk) + DcacheK$
 $Tfor(Kz) == Tfor(Kk)$
 $Texp(Kz) \geq Tret(Kz) + DcacheZ$

S(Kk) = (DNSKEY Dead, DS Dead)
P(Kk) = Forgotten ExpiredDS
S(Kz) = (DNSKEY Dead, RRSIG Dead)
P(Kz) = Forgotten Expired

4.3. Algorithm Rollover

When changing algorithms, it is possible that algorithms are added, removed or replaced. Adding and removing an algorithm follows the

same timings as enabling and disabling DNSSEC. Replacing an algorithm can be done with a STSS Double-Signature rollover or a KSK and ZSK Double-Signature Rollover at the same time. [MM: This needs more text, but I am awaiting the discussion about algorithm rollover and how to interpret section 2.2 of RFC 4035]

4.4. KSK-ZSK Split or Single Type Signing Scheme

When changing signing schemes, one should follow the timelines of the most restrictive signing scheme. The STSS signing scheme makes some rollover combinations unsuitable, thus it can be considered the most restricted signing scheme. In the case of moving to a KSK-ZSK Split, Ks is used as the successor key in the STSS rollover methods, and it now reflects both the successor ZSK and KSK. In the case of moving away from a KSK-ZSK Split, Kc is used as the predecessor key in the STSS rollover methods, and it now reflects both the predecessor ZSK and KSK. [MM: This could perhaps also use more explanation.]

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document does not introduce any new security issues beyond those already discussed in RFC4033 [RFC4033], RFC4034 [RFC4034], RFC4035 [RFC4035] and RFC5011 [RFC5011].

7. Acknowledgements

Special acknowledgments and gratitude go out to Stephen Morris, Johan Ihren and John Dickinson, the authors of [key-timing]. Significant parts of the text is taken from that document. Especially Section 3.2 and Section 3.3 are largely copied and adjusted to the new introduced terminology from this document.

Also, acknowledgements to Yuri Schaeffer, who brought to the attention the idea of key goals (Section 2.4) and whose discussions helped to shape this document.

8. Changelog

8.1. Changes with key-timing draft

This document builds further on [key-timing]. The most important changes with respect to that document are:

- Introduced the concept of Rollover Considerations (Speed vs Size vs Interactions), that causes the existence of different key rollover scenarios.
- Introduced the concept of Key Goals.
- Key States are unraveled to represent the status of each piece of information separately. Provides more flexibility. Used for combining rollover methods in a Single Type Signing Scheme.
- Four new Key States are introduced: Known, Safe, Forgotten and Expired, to represent whether information about the key exist in validator caches. The key states Ready and Dead are deprecated.
- Timelines for STSS Rollovers.
- Timelines for enabling and disabling DNSSEC.
- Text about policy rollover, such as algorithm rollover and changing signing schemes.

8.2. From -00 to -01

- Initial review Stephen Morris.
- Changed style, removed all first and second person style.
- Key Conditions are now called Key Properties.
- More detailed explanation on Key States Unraveled: Introduced Validation Components and Key Properties, described in different sections.
- Put the correct timeline figure in the section on STSS Double-DS rollover method.
- Review Marc Lampo, Stephen Morris
- Stephen provided text for the section on Key Rollover Stages

8.3. From -01 to -02

- Added a paragraph on document outline.
- Key Properties are now called Key States.
- Renamed Validation Component State 'Generated' to 'Hidden', renamed Key State 'Generated' to 'Uninformed'.

- Renamed Rollover Stages 'Activation' and 'Activated' to 'Transition' and 'Transited'.

9. References

9.1. Informative References

- [RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, September 2006.

9.2. Normative References

- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC 5011, September 2007.
- [dps-framework] Ljunggren, F., Eklund-Lowinder, A-M., and T. Okubo, "DNSSEC Policy & Practice Statement Framework", March 2011.
- [key-timing] Morris, S., Ihren, J., and J. Dickinson, "DNSSEC Key Timing Considerations", March 2011.

Appendix A. List of Symbols

[MM: To do]

Author's Address

Matthijs Mekking
NLnet Labs
Science Park 140
Amsterdam 1098 XG
The Netherlands

E-Mail: matthijs@nlnetlabs.nl

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

O. Gudmundsson
Shinkuro Inc.
February 14, 2014

Providing support for multiple namespaces to Internet protocols.
draft-ogud-appsawg-multiple-namespaces-00

Abstract

Over the years there have been various proposals as to use namespaces other than the DNS/IN namespace that is under ICANN administration. Some of these were simple DNS competitors others use different lookup technologies. In addition it is hard to supply different character encodings to DNS as each application needs to provide the translation between the encoding used and the format expected by DNS.

This draft proposes a new service layer to provide multiplexing of different namespaces into appropriate function calls.

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Not all Namespaces are equal	3
3. Illustrative Examples	3
3.1. Non DNS namespace	4
3.2. Different DNS class	4
4. IANA considerations	4
5. Security considerations	5
6. Acknowledgements	5
7. Informative References	5
Appendix A. Document history	5
Author's Address	5

1. Introduction

Most of us assume that there exists only one namespace on the Internet, the DNS in class IN [RFC1034]. But in practice there are multiple, in addition there are IPv4 and IPv6 address spaces, the Bonjour/MDNS local net namespace. Number of groups are proposing new namespaces that may or may not use DNS as underlying resolution protocol.

The "state of the art" in identifying what namespace is used is by using a postfix on the domain name, for example "ogud.onion" is a lookup in a distributed hash table [GNU]. As identified in this draft ALT TLD [ALT], proposing that such new uses be placed in a "squatters TLD" called .alt there are many drawbacks to using what looks like a DNS name to express a different lookup mechanism. One effect is the overhead of trying to locally stop queries to go to the internet[RFC6303] and special use registry exists[RFC6761]. Recently a draft [ReserveTLD] requested a number of allocations some of which are for namespaces that are not normal global DNS namespaces.

A further side effect of trying to fit new namespaces into the DNS namespace via TLD name or other trailing names is that developers then try to extend the resolution functions on the host OS to support multiple lookup mechanisms.

This draft proposes a radical solution of placing a "Namespace Identifier" at the front of the name and relies on new software to perform multiplexing between the different namespace lookup

mechanisms available on an host. This technique allows the host itself to reject unsupported lookups rather than leak them to DNS resolvers, or to authoritative servers. This is similar to URI but places an explicit namespace identifier on the "name".

At this point this is call for discussion rather than an proposed solution, all examples in this document are for illustration only. It is easy to get into adversarial discussions on this topic thus the draft tries to be non-confrontational, if the draft fails at that the editor apologizes.

1.1. Terminology

None at this time.

2. Not all Namespaces are equal

Right now the Internet has two universal namespace (DNS in class IN) and IPv4 addresses. There are number of minor namespaces in use such as .local for local networks, DNS class CHAOS for few applications.

DNS class field has not been widely used due to perceived difficulties in setting up new set of root servers for each class and propagate the information to the clients.

Using suffix TLD name on the other hand requires the hosts/applications/resolvers to know the suffix string and have the capability to resolve using the resolution process specified for that suffix.

It is not hard to imagine namespaces that work quite differently from DNS such as by creating long lived connections to the "servers" or exchange information via "modern" encodings like Json [RFC4627].

For these reasons it seems a better solution for multiple namespace existence to have OS's provide a Namespace Abstraction layer that applications call and each name space registers with the Abstraction layer the functions to use.

3. Illustrative Examples

The proposal for the "Meta-TLD" .gnu [GNU] specifies a distributed hash table to avoid centralized control of the namespace. This resolution is nothing like the DNS resolution so trying to extend functions like gethostbyname() to support multiple different lookup techniques is likely to have adverse effects on the reliability of the base OS functions.

3.1. Non DNS namespace

For a new namespace MARGIR (Icelandic for many) names are expressed like this:

```
https://MARGIR##ACDBU00CABU0R1FEL7V75RQE1G2GSVQM
application will call function
ret = namespace_lookup( "MARGIR##ACDBU00CABU0R1FEL7V75RQE1G2GSVQM",
                        TYPE_ADDR, &answer) ;
```

If the function returns error code "do not understand name space" the application knows this is something it can not use, there is no leak of information from the host. If the function returns success then the application can use the answer structure and proceed.

Underneath the namespace_lookup() function there are calls to various functions for each namespace, these functions are provided via libraries, and each namespace provides a binding to the standard calls such as:

```
Margir = {
    TYPE_ADDR = margir_lookup(A1, A2, &A3);
    TYPE_TXT  = margir_lookup(A1, 888, &A3);
    ....
}
```

3.2. Different DNS class

For a DNS lookup in a different class it is much easier to reuse the existing lookup functions but still a translation is useful.

```
https://CLASS666##bad.lucifer.
```

Translation table can be

```
CLASS666 = {
    TYPE_ADDR = gethostname(A1, CLASS_666, TYPE_A, &A3);
    TYPE_TXT  = query_by_type(A1, CLASS_666, TYPE_TXT, &A3);
    ....
}
```

4. IANA considerations

This document has no actions for IANA.

5. Security considerations

TBD. If the proposed work goes forward there are many difficult security issues that need to be addressed, the exact security issues depends on the actual proposal.

6. Acknowledgements

7. Informative References

- [ALT] Kumari, W. and A. Sullivan, "The ALT Special Use Top Level Domain", draft-wkumari-dnsop-alt-tld (work in progress), February 2014.
- [GNU] Grothoff, C., Wachs, M., Wolf, H., and J. Applebaum, "Special-Use Domain Names of Peer-to-Peer Systems", draft-iesg-special-use-p2p-names (work in progress), December 2013.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, July 2011.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.
- [ReserveTLD] Chapin, L. and M. McFadden, "Additional Reserved Top Level Domains", draft-chapin-additional-reserved-tlds (work in progress), January 2014.

Appendix A. Document history

[RFC Editor: Please remove this section before publication]

00 Initial version

Author's Address

Olafur Gudmundsson
Shinkuro Inc.
4922 Fairmont Av, Suite 250
Bethesda, MD 20814
USA

Email: ogud@ogud.com

dnsop
Internet-Draft
Intended status: Informational
Expires: November 19, 2015

W. Kumari
Google
A. Sullivan
Dyn
May 18, 2015

The ALT Special Use Top Level Domain
draft-wkumari-dnsop-alt-tld-06

Abstract

This document reserves a string (ALT) to be used as a TLD label in non-DNS contexts or for names that have no meaning in a global context. It also provides advice and guidance to developers developing alternate namespaces.

[Ed note: This document lives in GitHub at:
<https://github.com/wkumari/draft-wkumari-dnsop-alt-tld> . Issues and pull requests happily accepted.]

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 November 19, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements notation	2
1.2. Terminology	3
2. Background	3
3. The ALT namespace	4
4. Advice to developers	6
5. IANA Considerations	6
5.1. Domain Name Reservation Considerations	7
6. Security Considerations	8
7. Acknowledgements	8
8. References	8
8.1. Normative References	8
8.2. Informative References	8
Appendix A. Changes / Author Notes.	9
Authors' Addresses	10

1. Introduction

Many protocols and systems need to name entities. Names that look like DNS names (a series of labels separated with dots) have become common, even in systems that are not part of the global DNS.

This document provides a solution that may be more appropriate than [RFC6761] in many cases. RFC6761 specifies Special Use TLDs which should only be used in exceptional circumstances.

This document reserves the label "ALT" (short for "Alternate") as a Special Use Domain ([RFC6761]). This label is intended to be used as the final label (apart from the zero-length terminating label) to signify that the name is not rooted in the DNS, and that normal registration and lookup rules do not apply.

1.1. Requirements notation

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 [RFC2119].

1.2. Terminology

This document assumes familiarity with DNS terms and concepts. Please see [RFC1034] for background and concepts.

- o DNS context: The namespace anchored at the globally-unique DNS root. This is the namespace or context that "normal" DNS uses.
- o non-DNS context: Any other (alternate) namespace.
- o pseudo-TLD: A label that appears in a fully-qualified domain name in the position of a TLD, but which is not registered in the global DNS.
- o TLD: The last visible label in either a fully-qualified domain name or a name that is qualified relative to the root. See the discussion in Section 2.

2. Background

The DNS data model is based on a tree structure, and has a single root. Conventionally, a name immediately beneath the root is called a "Top Level Domain" or "TLD". TLDs usually delegate portions of their namespace to others, who may then delegate further. The hierarchical, distributed and caching nature of the DNS has made it the primary resolution system on the Internet.

Domain names are terminated by a zero-length label, so the root label is normally invisible. Truly fully-qualified names indicate the root label explicitly, thus: "an.example.tld.". Most of the time, names are written implicitly relative to the root, thus: "an.example.tld". In both of these cases, the TLD is the last label that is visible in presentation format -- in this example, the string "tld". (This little bit of pedantry is here because, in different contexts, people can use the term "fully-qualified domain name" to refer to either of these uses.) It is worth noting that the root label is present in the on-wire format of fully-qualified domain names, even if not displayed in the presentation form.

The success of the DNS makes it a natural starting point for systems that need to name entities in a non-DNS context, or that have no unique meaning in a global context. These name resolutions, therefore, occur in a namespace distinct from the DNS.

In many cases, these systems build a DNS-style tree parallel to the global DNS administered by IANA. They often use a pseudo-TLD to cause resolution in the alternate namespace, using browser plugins,

shims in the name resolution process, or simply applications that perform special handling of this alternate namespace.

In many cases, the creators of these alternate namespaces have chosen a convenient or descriptive string and started using it. These new strings are "alternate" strings and are not registered anywhere or part of the DNS. However they appear to be TLDs. Issues may arise if they are looked up in the DNS. These include:

- o User confusion: If someone emails a link of the form foo.bar.pseudo-TLD to someone who does not have the necessary software to resolve names in the pseudo-TLD namespace, the name will not resolve and the user may become confused.
- o Excess traffic hitting the DNS root: Lookups leak out of the pseudo-TLD namespace and end up hitting the DNS root nameservers.
- o Collisions: If the pseudo-TLD is eventually delegated from the root zone the behavior may be non-deterministic.
- o Lack of success for the user's original goal.

An alternate name resolution system might be specifically designed to provide confidentiality of the looked up name, and to provide a distributed and censorship resistant namespace. This goal would necessarily be defeated if the queries leak into the DNS, because the attempt to look up the name would be visible at least to the operators of root name servers.

3. The ALT namespace

In order to avoid the above issues, we reserve the ALT label. Unless the name desired is globally unique, has meaning on the global context and is delegated in the DNS, it should be considered an alternate namespace, and follow the ALT label scheme outlined below. The ALT label MAY be used in any domain name as a pseudo-TLD to signify that this is an alternate (non-DNS) namespace.

Alternate namespaces should differentiate themselves from other alternate namespaces by choosing a name and using it in the label position just before the pseudo-TLD (ALT). For example, a group wishing to create a namespace for Friends Of Olaf might choose the string "foo" and use any set of labels under foo.alt.

As they are in an alternate namespace, they have no significance in the regular DNS context and so should not be looked up in the DNS context. Unfortunately simply saying that "something should not happen" doesn't actually stop it from happening, so we need some

rules to guide implementors and operators. The ALT TLD is delegated to "new style" AS112 servers, and so recursive and stub resolvers will get NXDOMAIN for all queries.

1. Iterative resolvers SHOULD follow the advice in [RFC6303], Section 3.
2. The ALT TLD is delegated to "new style" AS112 nameservers ([I-D.ietf-dnsop-as112-dname]), which will return NXDOMAIN for all queries.

These rules are intended to limit how far unintentional queries (i.e. those not intended for the global DNS) flow.

Groups wishing to create new alternate namespaces SHOULD create their alternate namespace under a label that names their namespace, and under the ALT label. They SHOULD choose a label that they expect to be unique and, ideally, descriptive.

Currently deployed projects and protocols that are using pseudo-TLDs may decide to move under the ALT TLD, but this is not a requirement. Rather, the ALT TLD is being reserved so that future projects of a similar nature have a designated place to create alternate resolution namespaces that will not conflict with the regular DNS context.

A number of names other than .ALT were considered and discarded. In order for this technique to be effective the names need to continue to follow both the DNS format and conventions (a prime consideration for alternate name formats is that they can be entered in places that normally take DNS context names); this rules out using suffixes that do not follow the usual letter, digit, and hyphen label convention. Another proposal was that the ALT TLD instead be a reservation under .arpa. This was considered, but rejected for several reasons, including:

1. We wished this to make it clear that this is not in the DNS context, and .arpa clearly is.
2. The use of the string .ALT is intended to evoke the alt.* hierarchy in Usenet.
3. We wanted the string to be short and easily used.
4. A name underneath .arpa would consume at least five additional octets of the total 255 octets available in domain names, which could put pressure on applications that need long machine-generated names.

5. We are suggesting that the string .ALT get special treatment in resolvers, and shim software. We are concerned that using subdomains of an existing TLD (like .arpa) might end up with bad implementations misconfiguring / overriding the TLD itself and breaking .arpa.

There is a concern that if there were placed under .arpa, inexperienced nameserver operators may inadvertently cover .arpa. A more significant concern is that the scope of the issue if the query does leak, and the fact that this would then make the root of the alternate naming namespace a third level domain, and not a second one. A project may be willing to have a name of the form example.alt, but example.alt.arpa may be not look as good.

4. Advice to developers

Often, a subdomain of an existing, owned domain may suffice. When that is so, using a subdomain in the DNS is always preferable, and safest in terms of not risking misuse, duplications, or collisions. In the rare instance in which it is not desirable to have the name in the DNS, the .ALT namespace may be used.

In a number of cases the purpose of the alternate name resolution system is to provide confidentiality. For these systems the above advice is problematic. If the query for one of these names (for example harry.foo.example.com were to leak into the DNS, the query would hit the recursive resolver, and (assuming empty caches) would then hit the root, the .com name servers, the example.com name servers and then the foo.example.com nameservers. This means that the fact that a user is resolving harry.foo.example.com would be visible to a large number of people. Furthermore, the harry.foo.example.com nameservers become a good oracle to determine what names exist, and who is trying to reach them.

For projects that are very latency sensitive, or that desire to provide confidentiality, we recommend anchoring the alternate namespace under the .ALT TLD.

5. IANA Considerations

The IANA is requested to add the ALT string to the "Special-Use Domain Name" registry ([RFC6761], and reference this document. In addition, the "Locally Served DNS Zones" ([RFC6303]) registry should be updated to reference this document.

5.1. Domain Name Reservation Considerations

This section is to satisfy the requirement in Section 5 of RFC6761.

The domain "alt.", and any names falling within ".alt.", are special in the following ways:

1. Human users are expected to know that strings that end in .alt behave differently to normal DNS names. Users are expected to have applications running on their machines that intercept strings of the form <namespace>.alt and perform special handling of them. If the user tries to resolve a name of the form <namespace>.alt without the <namespace> plugin installed, the request will leak into the DNS, and receive a negative response.
2. Writers of application software that implement a non-DNS namespace are expected to intercept names of the form <namespace>.alt and perform application specific handling with them. Other applications are not intended to perform any special handling.
3. In general, writers of name resolution APIs and libraries do not need to perform special handling of these names. If developers of other namespaces implement their namespace through a "shim" or library, they will need to intercept and perform their own handling.
4. Caching DNS servers SHOULD recognize these names as special and SHOULD NOT, by default, attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve these names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.
5. Authoritative DNS servers SHOULD recognize these names as special and SHOULD, by default, generate immediate negative responses for all such queries, unless explicitly configured by the administrator to give positive answers for private-address reverse-mapping names.
6. DNS server operators SHOULD be aware that queries for names ending in .alt are not DNS names, and were leaked into the DNS context (for example, by a missing browser plugin). This information may be useful for support or debugging purposes.
7. DNS Registries/Registrars MUST NOT grant requests to register "alt" names in the normal way to any person or entity. These "alt" names are defined by protocol specification to be

nonexistent, and they fall outside the set of names available for allocation by registries/registrars.

6. Security Considerations

One of the motivations for the creation of the alt pseudo-TLD is that unmanaged labels in the managed root name space are subject to unexpected takeover if the manager of the root name space decides to delegate the unmanaged label.

The unmanaged and "registration not required" nature of labels beneath .ALT provides the opportunity for an attacker to re-use the chosen label and thereby possibly compromise applications dependent on the special host name.

7. Acknowledgements

The authors understand that there is much politics surrounding the delegation of a new TLD and thank the ICANN liaison in advance.

We would also like to thank Joe Abley, Mark Andrews, Marc Blanchet, John Bond, Stephane Bortzmeyer, David Cake, David Conrad, Patrik Faltstrom, Olafur Gudmundsson, Paul Hoffman, Joel Jaeggli, Ted Lemon, Edward Lewis, George Michaelson, Ed Pascoe, Arturo Servin, and Paul Vixie for feedback.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, July 2011.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

8.2. Informative References

- [I-D.ietf-dnsop-as112-dname]
Abley, J., Dickson, B., Kumari, W., and G. Michaelson,
"AS112 Redirection using DNAME", draft-ietf-dnsop-
as112-dname-06 (work in progress), November 2014.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -05 to -06

- o Incorporated comments from a number of people, including a number of suggestion heard at the IETF meeting in Dallas, and the DNSOP Interim meeting in May, 2015.
- o Removed the "Let's have an (optional) IANA registry for people to (opportunistically) register their string, if they want that option" stuff. It was, um, optional....

From -04 to -05

- o Went through and made sure that I'd captured the feedback received.
- o Comments from Ed Lewis.
- o Filled in the "Domain Name Reservation Considerations" section of RFC6761.
- o Removed examples from .Onion.

From -03 to -04

- o Incorporated some comments from Paul Hoffman

From -02 to -03

- o After discussions with chairs, made this much more generic (not purely non-DNS), and some cleanup.

From -01 to -02

- o Removed some fluffy wording, tightened up the language some.

From -00 to -01.

- o Fixed the abstract.
- o Recommended that folk root their non-DNS namespace under a DNS namespace that they control (Joe Abley)

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Andrew Sullivan
Dyn
150 Dow Street
Manchester, NH 03101
US

Email: asullivan@dyn.com

dnsext
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

P. Wouters
Red Hat
February 14, 2014

Chain Query requests in DNS
draft-wouters-edns-chain-query-00

Abstract

This document defines an EDNS0 extension that can be used by a DNSSEC enabled Recursive Nameserver configured as a forwarder to send a single DNS query requesting to receive a complete validation path along with the regular DNS answer, without the need to rapid-fire many UDP requests in an attempt to attain a low latency.

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Notation	3
2. Terminology	3
3. Overview	4
4. Option Format	5
5. Protocol Description	5
5.1. Discovery of Support	5
5.2. Generating a Query	5
5.3. Generating a Response	6
5.4. Sending the Option	7
6. Protocol Considerations	7
6.1. DNSSEC Considerations	7
6.2. NS record Considerations	7
6.3. TCP Session Management	8
6.4. Non-Clean Paths	8
6.5. Anycast Considerations	8
7. Implementation Status	9
8. Security Considerations	9
8.1. Amplification Attacks	9
9. Examples	10
9.1. Simple Query for example.com	10
9.2. Out-of-path query for example.com	12
9.3. non-existent data	12
10. IANA Considerations	13
10.1. EDNS0 option code for edns-chain-query	13
11. Acknowledgements	14
12. Normative References	14
Author's Address	15

1. Introduction

Traditionally, clients operate in stub-mode for DNS. For each DNS question the client needs to resolve, it sends a single query to an upstream DNS resolver to obtain a single DNS answer. When DNSSEC [RFC4033] is deployed on such clients, validation requires that the client obtains all the (intermediate) information from the DNS root down to the queried-for hostname so it can perform DNSSEC validation on the complete chain of trust. This process increases the number of DNS queries and answers required, and thus increases the latency before a validated DNS answer has been obtained.

Currently, applications use a rapid-fire approach to send out many UDP requests concurrently. This requires more resources on the DNS client with respect to state (cpu, memory, battery) and bandwidth. There is also no guarantee that the initial burst of UDP questions will result in all the records required for DNSSEC validation, and

more round trips could be required depending on the resulting DNS answers. This especially affects high-latency links.

This document specifies an EDNS0 extension that allows a validating recursive name server running as a forwarder to request another recursive name server for a DNS chain answer using one DNS query/answer pair. This reduces the number of round-trip times ("RTT") to two. If combined with [TCP-KEEPALIVE] there is only 1 RTT. While the upstream DNS resolver still needs to perform all the individual queries required for the complete answer, it usually has a much bigger cache and does not experience significant slowdown from last-mile latency.

This EDNS0 extension allows the Forwarder to indicate which part of the DNS hierarchy it already contains in its cache. This reduces the amount of data required to be transferred and reduces the work the upstream Resolving Nameserver has to perform.

This EDNS0 extension is only intended for Forwarders. It can (and should be) ignored by Authoritative Nameservers and by Recursive Nameservers that do not support this EDNS0 option.

1.1. Requirements Notation

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 [RFC2119].

2. Terminology

Stub Resolver: A simple DNS protocol implementation on the client side as described in [RFC1034] section 5.3.1.

Authoritative Nameserver: A nameserver that has authority over one or more DNS zones. These are normally not contacted by clients directly but by Recursive Resolvers. Described in [RFC1035] chapter 6.

Recursive Resolver: A nameserver that is responsible for resolving domain names for clients by following the domain's delegation chain, starting at the root. Recursive Resolvers frequently use caches to be able to respond to client queries quickly. Described in [RFC1035] chapter 7.

Validating Resolver: A recursive nameserver that also performs DNSSEC [RFC4033] validation.

Forwarder: A Recursive Resolver that is using another (upstream) Recursive Resolver instead of querying Authoritative Nameservers directly. It still performs validation.

3. Overview

When DNSSEC is deployed on the client, it can no longer delegate all DNS work to the upstream Resolving Nameserver. Obtaining just the DNS answer itself is not enough to validate that answer using DNSSEC. For DNSSEC validation, the client requires a locally running validating DNS server configured as Resolving Nameserver so it can confirm DNSSEC validation of all intermediary DNS answers. It can configure itself as a Forwarder if the DHCP server has indicated that one or more Resolving Nameservers are available. Regardless, generating the required queries for validation adds a significant delay in answering the DNS question of the locally running application. The application has to wait while the Forwarder on the client is querying for all the intermediate work. Each round-trip adds to the total time waiting on DNS resolving to complete. This makes DNSSEC resolving impractical on networks with a high latency.

The edns-chain-query option allows the client to request all intermediate DNS data it requires to resolve and validate a particular DNS answer in a single round-trip DNS query and answer.

Servers answering with chain query data exceeding 512 bytes should ensure that the transport is TCP or source IP address verified UDP. See Section 8. This avoids abuse in DNS amplification attacks.

The format of this option is described in Section 4.

As described in Section 5.3, a recursive nameserver could use this EDNS0 option to include additional data required by the client in the Authority Section of the DNS answer packet when using a source IP verified transport. The Answer Section remains unchanged from a traditional DNS answer and contains the answer and related DNSSEC entries.

An empty edns-chain-query EDNS0 option MAY be sent over any transport as a discovery method. A DNS server receiving such an empty edns-chain-query option SHOULD add an empty edns-chain-query option in its answer to indicate that it supports edns-chain-query for source IP address verified transports.

The mechanisms provided by edns-chain-query raise various security related concerns, related to the additional work, bandwidth, amplification attacks as well as privacy issues with the cache. These concerns are described in Section 8.

In this option value, the Forwarder sets the last known entry point in the chain - furthest from the root - that it already has a DNSSEC validated (secure or not) answer for in its cache. The upstream Recursive Resolver does not need to include any part of the chain from the root down to this option's FQDN. A complete example is described in Section 9.

Depending on the size of the labels of the last known entry point value, a DNS Query packet could be arbitrarily large. If using the last known entry point would result in a query size of more than 512 bytes, the last known entry point should be replaced with its parent entry until the query size would be 512 bytes or less.

5.3. Generating a Response

When a query containing a non-zero edns-chain-query option is received over a TCP connection from a Forwarder, the upstream Recursive Resolver supporting edns-chain-query MAY respond by confirming that it is returning a DNS Query Chain. To do so, it MUST set the edns-chain-query option with an OPTION-LENGTH of zero to indicate the DNS answer contains a Chain Query. It extends the Authority Section for the DNS answer packet with the required DNS RRs resulting in an Authority Section that contains a complete chain of DNS RRs that start with the first chain element below the received Last Known Query Name up to and including the NS and DS RRs that represent the zone cut (authoritative servers) of the QNAME. The actual DNS answer to the question in the Query Section is placed in the DNS Answer Section identical to traditional DNS answers. If the received query has the DNSSEC OK flag set, all required DNSSEC related records must be added to their appropriate sections. This includes records required for proof of non-existence of regular and/or wildcard records, such as NSEC or NSEC3 records.

Recursive Resolvers that have not implemented or enabled support for the edns-chain-query option, or are otherwise unwilling to perform the additional work for a Chain Query due to work load, may safely ignore the option in the incoming queries. Such a server MUST NOT include an edns-chain-query option when sending DNS answer replies back, thus indicating it is not able to support Chain Queries at this time.

Requests with wrongly formatted options (i.e. bogus FQDN) MUST be rejected and a FORMERR response must be returned to the sender, as described by [RFC2671], Transport Considerations.

Requests resulting in chains that the receiving resolver is unwilling to serve can be rejected by sending a REFUSED response to the sender, as described by [RFC2671], Transport Considerations. This refusal

can be used for chains that would be too big or chains that would reveal too much information considered private.

At any time, a DNS server that has determined that it is running low on resources can refuse to acknowledge a Chain Query by omitting the edns-chain-query option. It may do so even if it conveyed support to a DNS client previously. If [TCP-KEEPAALIVE] is used, it may even change its support for edns-chain-query within the same TCP session.

If the DNS request results in an CNAME or DNAME for the Answer Section, the DNS server MUST return these records in the Answer Section similar to regular DNS processing. The CNAME or DNAME target MAY be placed in the Additional Section only if all supporting records for DNSSEC validation of the CNAME or DNAME target is also added to the Authority Section.

In any case, the response from the receiving resolver to the client resolver MUST NOT contain the edns-chain-query option if none was present in the client's resolver original request.

5.4. Sending the Option

When edns-chain-query is available, the downstream Resolving Nameserver can adjust its query strategy based on the desired queries and its cache contents.

A Forwarder can request the edns-chain-query option with every outgoing DNS query. However, it is RECOMMENDED that Forwarders remember which upstream Resolving Nameservers did not return the option (and additional data) with their response. The Forwarder SHOULD fallback to regular DNS for subsequent queries to those Recursive Nameservers. It MAY switch to another Resolving Nameserver that does support the edns-chain-query option or try again later to see if the server has become less loaded and is now willing to answer with Query Chains.

6. Protocol Considerations

6.1. DNSSEC Considerations

The presence or absence of an OPT resource record containing an edns-chain-query option in a DNS query does not change the usage of those resource records and mechanisms used to provide data origin authentication and data integrity to the DNS, as described in [RFC4033], [RFC4034] and [RFC4035].

6.2. NS record Considerations

edns-chain-query responses MUST include the NS RRset from the child zone, which includes DNSSEC RRSIG records required for validation.

When a DNSSEC chain is supplied via edns-chain-query, the Forwarder no longer requires to use the NS RRset, as it can construct the validation path via the DNSKEY and DS RRsets without using the NS RRset. However, it is preferred that the Forwarder can populate its cache with this information regardless, to avoid requiring queries in the future just to obtain the missing NS records. This can happen on a roaming device that needs to switch from using a DHCP obtained DNS server as forwarder to running in full autonomous resolver mode, for example when the DHCP obtained DNS server is broken in some way.

6.3. TCP Session Management

It is recommended that TCP Chain Queries are used in combination with [TCP-KEEPAALIVE].

Both DNS clients and servers are subject to resource constraints which will limit the extent to which TCP Chain Queries can be executed. Effective limits for the number of active sessions that can be maintained on individual clients and servers should be established, either as configuration options or by interrogation of process limits imposed by the operating system.

In the event that there is greater demand for TCP Chain Queries than can be accommodated, DNS servers may stop advertising the edns-query-chain option in successive DNS messages. This allows, for example, clients with other candidate servers to query to establish new TCP sessions with different servers in expectation that those servers might still allow TCP Chain Queries.

6.4. Non-Clean Paths

Many paths between DNS clients and servers suffer from poor hygiene, limiting the free flow of DNS messages that include particular EDNS0 options, or messages that exceed a particular size. A fallback strategy similar to that described in [RFC6891] section 6.2.2 SHOULD be employed to avoid persistent interference due to non-clean paths.

6.5. Anycast Considerations

DNS servers of various types are commonly deployed using anycast [RFC4786].

Successive DNS transactions between a client and server using UDP transport may involve responses generated by different anycast nodes, and the use of anycast in the implementation of a DNS server is

effectively undetectable by the client. The edns-chain-query option SHOULD NOT be included in responses using UDP transport from servers provisioned using anycast unless all anycast server nodes are capable of processing the edns-query-chain option.

Changes in network topology between clients and anycast servers may cause disruption to TCP sessions making use of edns-chain-query more often than with TCP sessions that omit it, since the TCP sessions are expected to be longer-lived. Anycast servers MAY make use of TCP multipath [RFC6824] to anchor the server side of the TCP connection to an unambiguously-unicast address in order to avoid disruption due to topology changes.

7. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

[While there is some interest, no work has started yet]

8. Security Considerations

8.1. Amplification Attacks

Chain Queries can potentially send very large DNS answers. Attackers could abuse this using spoofed source IP addresses to inflict large Distributed Denial of Service attacks using query-chains as an amplification vector in their attack. While TCP is not vulnerable for this type of abuse, the UDP protocol is vulnerable to this.

A recursive nameserver MUST NOT return Query Chain answers to clients over UDP without source IP address verification, for instance using [EASTLAKE-COOKIES]. A recursive nameserver SHOULD signal support in response to a Query Chain request over UDP by responding using a zero-length edns-chain-query option over UDP even without source IP address verification.

9. Examples

9.1. Simple Query for example.com

1. A web browser on a client machine asks the Forwarder running on localhost to resolve the A record of "www.example.com." by sending a regular DNS UDP query on port 53 to 127.0.0.1.
2. The Forwarder on the client machine checks its cache, and notices it already has a DNSSEC validated entry of "com." in its cache. This includes the DNSKEY RRset with its RRSIG records. In other words, according to its cache, ".com" is DNSSEC validated as "secure" and can be used to continue a DNSSEC validated chain on.
3. The Forwarder on the client opens a TCP connection to its upstream Recursive Resolver on port 53. It adds the edns-chain-query option as follows:
 - * Option-code, set to [TBD]
 - * Option-length, set to 0x00 0x04
 - * Last Known Query Name set to "com."
4. The upstream Recursive Resolver receives a DNS query over TCP with the edns-chain-query Last Known Query Name set to "com.". After accepting the query it starts constructing a DNS reply packet.
5. The upstream Recursive Resolver performs all the regular work to ensure it has all the answers to the query for the A record of "www.example.com.". It does so without using the edns-chain-query option - unless it is also configured as a Forwarder. The answer to the original DNS question could be the actual A record, the DNSSEC proof of non-existence, or an insecure NXDOMAIN response.
6. The upstream Recursive Resolver adds the edns-chain-query option to the DNS answer reply as follows:

- * Option-code, set to [TBD]
 - * Option-length, set to 0x00 0x00
 - * The Last Known Query Name is omitted (zero length)
7. The upstream Recursive Resolver constructs the DNS Authority Section and fills it with:
- * The DS RRset for "example.com." and its corresponding RRSIGs (made by the "com." DNSKEY(s))
 - * The DNSKEY RRset for "example.com." and its corresponding RRSIGs (made by the "example.com" DNSKEY(s))
 - * The authoritative NS RRset for "example.com." and its corresponding RRSIGs (from the child zone)

If the answer does not exist, and the zone uses DNSSEC, it also adds the proof of non-existence, such as NSEC or NSEC3 records, to the Authority Section.

8. The upstream Recursive Resolver constructs the DNS Answer Section and fills it with:
- * The A record of "www.example.com." and its corresponding RRSIGs

If the answer does not exist (no-data or NXDOMAIN), the Answer Section remains empty. For the NXDOMAIN case, the RCode of the DNS answer packet is set to NXDOMAIN. Otherwise it remains NOERROR.

9. The upstream Recursive Resolver returns the DNS answer over the existing TCP connection. When all data is sent, it SHOULD keep the TCP connection open to allow for additional incoming DNS queries - provided it has enough resources to do so.
10. The Forwarder receives the DNS answer. It processes the Authority Section and the Answer Section and places the information in its local cache. It ensures that no data is accepted into the cache without having proper DNSSEC validation. It MAY do so by looping over the entries in the Authority and Answer Sections. When an entry is validated for its cache, it is removed from the processing list. If an entry cannot be validated it is left in the process list. When the end of the list is reached, the list is processed again until either all entries are placed in the cache, or the remaining items cannot

be placed in the cache due to lack of validation. Those entries are then disgarded.

11. If the cache contains a valid answer to the application's query, this answer is returned to the application via a regular DNS answer packet. This packet MUST NOT contain an edns-chain-query option. If no valid answer can be returned, normal error processing is done. For example, an NXDOMAIN or an empty Answer Section could be returned depending on the error condition.

9.2. Out-of-path query for example.com

A Recursive Resolver receives a query for the A record for example.com. It includes the edns-chain-query option with the following parameters:

- o Option-code, set to [TBD]
- o Option-length, set to 0x00 0x0D
- o The Last Known Query Name set to 'unrelated.ca.'

As there is no chain that leads from "unrelated.ca." to "example.com", the Resolving Nameserver answers with RCODE "FormErr". It includes the edns-chain-query with the following parameters:

- o Option-code, set to [TBD]
- o Option-length, set to 0x00 0x00
- o The Last Known Query Name is ommited (zero length)

9.3. non-existent data

A Recursive Resolver receives a query for the A record for "ipv6.toronto.redhat.ca". It includes the edns-chain-query option with the following parameters:

- o Option-code, set to [TBD]
- o Option-length, set to 0x00 0x03
- o The Last Known Query Name set to 'ca.'

Using regular UDP queries towards Authoritative Nameservers, it locates the NS RRset for "toronto.redhat.ca.". When querying for the A record it receives a reply with RCODE "NoError" and an empty Answer Section. The Authority Section contains NSEC3 and RRSIG records proving there is no A RRtype for the QNAME "ipv6.toronto.redhat.ca".

The Recursive Resolver constructs a DNS reply with the following edns-chain-query option parameters:

- o Option-code, set to [TBD]
- o Option-length, set to 0x00 0x00
- o The Last Known Query Name is omitted (zero length)

The RCODE is set to "NoError". The Authority Section is filled in with:

- o The DS RRset for "redhat.ca." plus RRSIGs
- o The DNSKEY RRset for "redhat.ca." plus RRSIGs
- o The NS RRset for "redhat.ca." plus RRSIGs (eg ns[01].redhat.ca)
- o The A RRset for "ns0.redhat.ca." and "ns1.redhat.ca." plus RRSIGs
- o The DS RRset for "toronto.redhat.ca." plus RRSIGs
- o The NS RRset for "toronto.redhat.ca." plus RRSIGs (eg ns[01].toronto.redhat.ca)
- o The DNSKEY RRset for "toronto.redhat.ca." plus RRSIGs
- o The A RRset and/or AAAA RRset for "ns0.toronto.redhat.ca." and "ns1.toronto.redhat.ca." plus RRSIGs
- o The NSEC record for "ipv6.toronto.redhat.ca." (proves what RRTYPES do exist, does not include A)
- o The NSEC record for "toronto.redhat.ca." (proves no wildcard exists)

The Answer Section is empty. The RCode is set to NOERROR.

10. IANA Considerations

10.1. EDNS0 option code for edns-chain-query

IANA has assigned option code [TBD] in the "DNS EDNS0 Option Codes (OPT)" registry to edns-chain-query.

11. Acknowledgements

Andrew Sullivan pointed out that we do not need any new data formats to support DNS chains. Olafur Gudmundsson ensured the RRsets are returned in the proper Sections.

12. Normative References

[EASTLAKE-COOKIES]

Eastlake, Donald., "Domain Name System (DNS) Cookies", draft-eastlake-dnsext-cookies (work in progress), January 2014.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, December 2006.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.
- [TCP-KEEPALIVE] Wouters, P., "The edns-tcp-keepalive EDNS0 Option", draft-wouters-edns-tcp-keepalive (work in progress), February 2014.

Author's Address

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

P. Wouters
Red Hat
J. Abley
Dyn, Inc.
February 14, 2014

The edns-tcp-keepalive EDNS0 Option
draft-wouters-edns-tcp-keepalive-01

Abstract

DNS messages between clients and servers may be received over either UDP or TCP. UDP transport involves keeping less state on a busy server, but can cause truncation and retries over TCP. Additionally, UDP can be exploited for reflection attacks. Using TCP would reduce retransmits and amplification. However, clients are currently limited in their use of the TCP transport as RFC 5966 suggests closing idle TCP sessions "in the order of seconds", making use of TCP only suitable for individual queries generated as a fallback protocol for truncated UDP answers.

This document defines an EDNS0 option ("edns-tcp-keepalive") that allows DNS clients and servers to signal their respective readiness to conduct multiple DNS transactions over individual TCP sessions. This signalling facilitates a better balance of UDP and TCP transport between individual clients and servers, reducing the impact of problems associated with UDP transport and allowing the state associated with TCP transport to be managed effectively with minimal impact on the DNS transaction time.

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Notation	4
3. The edns-tcp-keepalive Option	4
3.1. Option Format	4
3.2. Use by DNS Clients	5
3.2.1. Sending Queries	5
3.2.2. Receiving Responses	5
3.3. Use by DNS Servers	6
3.3.1. Receiving Queries	6
3.3.2. Sending Responses	6
3.4. TCP Session Management	7
3.5. Non-Clean Paths	7
3.6. Anycast Considerations	7
4. Security Considerations	8
5. IANA Considerations	8
6. Acknowledgements	8
7. References	8
7.1. Normative References	8
7.2. Informative References	9
Appendix A. Editors' Notes	9
A.1. Venue	9
A.2. Abridged Change History	9
A.2.1. draft-wouters-edns-tcp-keepalive-00	9
Authors' Addresses	9

1. Introduction

DNS messages between clients and servers may be received over either UDP or TCP [RFC1035]. Historically, DNS clients used API's that only facilitated sending and receiving a single query over either UDP or TCP. New APIs and deployment of DNSSEC validating resolvers on hosts

that in the past were using stub resolving only is increasing the DNS client base that prefer using long lived TCP connections. Long-lived TCP connections can result in lower request latency than the case where UDP transport is used and truncated responses are received, since clients that have fallen back to TCP transport in response to a truncated response typically only uses the TCP session for a single (request, response) pair, continuing with UDP transport for subsequent queries.

UDP transport is stateless, and hence presents a much lower resource burden on a busy DNS server than TCP. An exchange of DNS messages over UDP can also be completed in a single round trip between communicating hosts, resulting in optimally-short transaction times. UDP transport is not without its risks, however.

A single-datagram exchange over UDP between two hosts can be exploited to enable a reflection attack on a third party. Mitigation of such attacks on authoritative-only servers is possible using an approach known as Response Rate-Limiting [RRL], an approach designed to minimise the frequency at which legitimate responses are discarded by truncating responses that appear to be motivated by an attacker, forcing legitimate clients to re-query using TCP transport.

[RFC1035] specified a maximum DNS message size over UDP transport of 512 bytes. Deployment of DNSSEC [RFC4033] and other protocols subsequently increased the observed frequency at which responses exceed this limit. EDNS0 [RFC6891] allows DNS messages larger than 512 bytes to be exchanged over UDP, with a corresponding increased incidence of fragmentation. Fragmentation is known to be problematic in general, and has also been implicated in increasing the risk of cache poisoning attacks.

The use of TCP transport does not suffer from the risks of fragmentation nor reflection attacks. However, TCP transport as currently deployed has expensive overhead.

The overhead of the three-way TCP handshake for a single DNS transaction is substantial, increasing the transaction time for a single (request, response) pair of DNS messages from 1 x RTT to 2 x RTT. There is no such overhead for a session that is already established, however, and the overall impact of the TCP setup handshake when the resulting session is used to exchange N DNS message pairs over a single session, $(1 + N)/N$, approaches unity as N increases.

(It should perhaps be noted that the overhead for a DNS transaction over UDP truncated due to RRL is 3x RTT, higher than the overhead imposed on the same transaction initiated over TCP.)

With increased deployment of DNSSEC and new RRtypes containing application specific cryptographic material, there is an increase in the prevalence of truncated responses received over UDP with fallback to TCP.

The use of TCP transport requires considerably more state to be retained on DNS servers. If a server is to perform adequately with a significant query load received over TCP, it must manage its available resources to ensure that all established TCP sessions are well-used, and that those which are unlikely to be used for the exchange of multiple DNS messages are closed promptly.

This document proposes a signalling mechanism between DNS clients and servers that provides a means to better balance the use of UDP and TCP transport, reducing the impact of problems associated with UDP whilst constraining the impact of TCP on response times and server resources to a manageable level.

The reduced overhead of this extension adds up significantly when combined with other edns extensions, such as [CHAIN-QUERY]. The combination of these two EDNS extensions make it possible for hosts on high-latency mobile networks to natively perform DNSSEC validation.

2. Requirements Notation

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 [RFC2119].

3. The edns-tcp-keepalive Option

This document specifies a new EDNS0 [RFC6891] option, edns-tcp-keepalive, which can be used by DNS clients and servers to signal a willingness to keep an (idle) TCP session open for a certain amount of time to conduct future DNS transactions. This specification does not distinguish between different types of DNS client and server in the use of this option.

3.1. Option Format

The edns-tcp-keepalive option is encoded as follows:

1	2	3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1		
+-----+-----+-----+		
! OPTION-CODE !	! OPTION-LENGTH !	
+-----+-----+-----+		

```
|                                     TIMEOUT                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

where:

OPTION-CODE: the EDNS0 option code assigned to edns-tcp-keepalive,
[TBD]

OPTION-LENGTH: the value 2;

TIMEOUT: a timeout value for the TCP connection, specified in
seconds, encoded in network byte order.

3.2. Use by DNS Clients

3.2.1. Sending Queries

DNS clients MAY include the edns-tcp-keepalive option in queries sent using UDP transport to signal their general ability to use individual TCP sessions for multiple DNS transactions with a particular server.

DNS clients MAY include the edns-tcp-keepalive option in the first query sent to a server using TCP transport to signal their desire that that specific TCP session be used for multiple DNS transactions.

Clients MAY specify a TIMEOUT value that is representative of the minimum expected time an individual TCP session should remain established for it to be used by multiple DNS transactions.

In the case where there are multiple candidate servers available to service a particular transaction, clients MAY include data associated with all servers when computing a TIMEOUT value to be signalled to any one of those servers.

If the client has insufficient data to be able to provide a meaningful estimate, the TIMEOUT value MUST be set to zero.

3.2.2. Receiving Responses

A DNS client that receives a response using UDP transport that includes the edns-tcp-keepalive option MAY record the presence of the option and the associated TIMEOUT value, and use that information as part of its server selection algorithm in the case where multiple candidate servers are available to service a particular query.

A DNS client that receives a response using TCP transport that includes the edns-tcp-keepalive option MAY keep the existing TCP session open.

A DNS client that receives a response that includes the edns-tcp-keepalive option with a TIMEOUT value of 0 is allowed to keep the TCP connection open indefinitely.

3.3. Use by DNS Servers

3.3.1. Receiving Queries

A DNS server that receives a query using UDP transport that includes the edns-tcp-keepalive option MAY record the presence of the option for statistical purposes, but should not otherwise modify its usual behaviour in sending a response.

A DNS server that receives a query using TCP transport that includes the edns-tcp-keepalive option SHOULD extend the timeout normally associated with TCP sessions if resources permit, using the TIMEOUT value supplied by the client as a guide.

3.3.2. Sending Responses

DNS servers MAY include the edns-tcp-keepalive option in responses sent using UDP transport to signal their general ability to use individual TCP sessions for multiple DNS transactions with a particular server. The TIMEOUT value should be indicative of what a client might expect if it was to open a TCP session with the server and receive a response with the edns-tcp-keepalive option present. The DNS server MAY omit including the edns-tcp-keepalive option if it is running too low on resources to service more TCP keepalive sessions.

DNS servers MAY include the edns-tcp-keepalive option in responses sent using TCP transport to signal their ability to use that specific session to exchange multiple DNS transactions. Servers MUST specify the TIMEOUT value that is currently associated with the TCP session. It is reasonable for this value to change according to local resource constraints. The DNS server MAY omit including the edns-tcp-keepalive option if it deems its local resources are too low to service more TCP keepalive sessions.

3.4. TCP Session Management

Both DNS clients and servers are subject to resource constraints which will limit the extent to which TCP sessions can persist. Effective limits for the number of active sessions that can be maintained on individual clients and servers should be established, either as configuration options or by interrogation of process limits imposed by the operating system.

In the event that there is greater demand for TCP sessions than can be accommodated, servers may reduce the TIMEOUT value signalled in successive DNS messages to avoid abrupt termination of a session. This allows, for example, clients with other candidate servers to query to establish new TCP sessions with different servers in expectation that an existing session is likely to be closed, or to fall back to UDP.

DNS clients and servers MAY close a TCP session at any time in order to manage local resource constraints. The algorithm by which clients and servers rank active TCP sessions in order to determine which to close is not specified in this document.

3.5. Non-Clean Paths

Many paths between DNS clients and servers suffer from poor hygiene, limiting the free flow of DNS messages that include particular EDNS0 options, or messages that exceed a particular size. A fallback strategy similar to that described in [RFC6891] section 6.2.2 SHOULD be employed to avoid persistent interference due to non-clean paths.

3.6. Anycast Considerations

DNS servers of various types are commonly deployed using anycast [RFC4786].

Successive DNS transactions between a client and server using UDP transport may involve responses generated by different anycast nodes, and the use of anycast in the implementation of a DNS server is effectively undetectable by the client. The edns-tcp-keepalive option SHOULD NOT be included in responses using UDP transport from servers provisioned using anycast unless all anycast server nodes are capable of processing the edns-tcp-keepalive option.

Changes in network topology between clients and anycast servers may cause disruption to TCP sessions making use of edns-tcp-keepalive more often than with TCP sessions that omit it, since the TCP sessions are expected to be longer-lived. Anycast servers MAY make use of TCP multipath [RFC6824] to anchor the server side of the TCP

connection to an unambiguously-unicast address in order to avoid disruption due to topology changes.

4. Security Considerations

The edns-tcp-keep-alive option can potentially be abused to request large numbers of sessions in a quick burst. When a Nameserver detects abusive behaviour, it SHOULD immediately close the TCP connection and free all buffers used.

This section needs more work. As usual.

5. IANA Considerations

The IANA is directed to assign an EDNS0 option code for the edns-tcp-keepalive option from the DNS EDNS0 Option Codes (OPT) registry as follows:

Value	Name	Status	Reference
[TBA]	edns-tcp-keepalive	Optional	[This document]

6. Acknowledgements

The authors acknowledge the contributions of Ray Bellis, Jinmei TATUYA and Mark Andrews.

7. References

7.1. Normative References

- [CHAIN-QUERY] Wouters, P., "chain Query requests in DNS", draft-wouters-edns-chain-query (work in progress), February 2014.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, December 2006.
- [RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, August 2010.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.

7.2. Informative References

- [RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1-Draft1, April 2012.

Appendix A. Editors' Notes

A.1. Venue

An appropriate venue for discussion of this document is dnsext@ietf.org.

A.2. Abridged Change History

A.2.1. draft-wouters-edns-tcp-keepalive-00

Initial draft.

Authors' Addresses

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com