Domain Name System Operations   T. Finch
Internet-Draft   University of Cambridge
Obsoletes: 2317 (if approved)   November 10, 2015
Updates: 2136 (if approved)
Intended status: Standards Track
Expires: May 13, 2016

Classless IN-ADDR.ARPA delegation and dynamic reverse DNS UPDATE
draft-fanf-dnsop-rfc2317bis-01

Abstract

This memo describes how to do IN-ADDR.ARPA delegation on any non-octet boundary, and how to consolidate reverse DNS for multiple address blocks into one zone.

It also clarifies the behaviour of dynamic reverse DNS UPDATE clients.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Since the introduction of classless inter-domain routing (CIDR), it has become common to assign IPv4 address space on non-octet boundaries. This memo describes how to do IN-ADDR.ARPA delegation on any non-octet boundary. There are two complementary methods, using CNAME records for long prefixes (greater than 24 bits) and using DNAME records for shorter prefixes.

The CNAME method (Section 4) makes it possible to assign IP address blocks with prefixes longer than 24 bits, covering fewer than 256 addresses, without losing the ability to delegate authority for the corresponding IN-ADDR.ARPA mappings. This method is fully compatible with the original DNS lookup mechanisms specified in [RFC1034], i.e. there is no need to modify the lookup algorithm used, and there should be no need to modify any software which does DNS lookups.

For shorter prefixes IN-ADDR.ARPA space is usually delegated on octet boundaries, which can lead to a proliferation of zones, for instance a /17 assignment requires 128 delegations. The DNAME method (Section 6) makes it possible to reduce the number of zones to match the number of address space assignments. Although DNAME records [RFC6672] are an extension to the original DNS specification, they are sufficiently widely supported to make this method feasible. There is a discussion of DNAME deployment considerations in [RFC7535] section 6.

These methods can also be used to consolidate multiple address blocks into a single DNS zone. (Section 8.) This reduces the administrative overhead of managing delegations; for instance it reduces the need to update DS records when DNSSEC keys are rolled.

While these methods interoperate well with DNS resolvers, they require some care from dynamic DNS UPDATE clients that are trying to change IN-ADDR.ARPA mappings. The client needs to follow the CNAME and/or DNAME redirections so that its UPDATE request changes the canonical PTR record without disrupting the redirections. (Section 9.)

1.1. Terminology

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

Examples use the "example" special-use domain name [RFC6761], the example networks 192.0.2.0/24 [RFC5735] and 2001:db8::/32 [RFC5156], and to illustrate shorter prefixes, the private network 10.0.0.0/8

[RFC1918].

2. DNS master file $GENERATE directive

The examples in this memo are written using DNS master file syntax as specified in [RFC1035] section 5. Some examples use the $GENERATE extension, which allows you to generate multiple resource records based on a numeric range and a template. The $GENERATE extension is well-known but it is not supported by all DNS master file readers.

The syntax of the $GENERATE directive is:

```
$GENERATE range domain [ttl] [class] type rdata [comment]
```

The directive and numeric range are followed by a template resource record. The ttl, class, type, and comment fields are in standard master file format.

The range is a pair of unsigned decimal numbers separated by a "-", and the left-hand (first) number is less than or equal to the right-hand (last) number. The $GENERATE directive expands an instance of the template for each number in the range including both endpoints.

```
range  =  1*DIGIT "-" 1*DIGIT  ; first-last
```

The owner domain of the template is a normal domain name, except that where "$" occurs it is replaced by the generated number. A literal "$" can be included by escaping it with a backslash, like "\$", or doubling it, like "$$".

The rdata is expanded in the same way as the domain. If the rdata includes white space (as in MX records, for example) then the whole rdata must be quoted. If the rdata needs to be quoted, then it must be quoted twice.

```
$GENERATE 0-1 $.text.example. TXT "\"slightly arcane\""
```

Extended versions of the $GENERATE directive allow you to modify a substitution by following the "$" with a clause in braces like "${...}". These modifiers are not described here.

For example, you can use the $GENERATE directive to populate the reverse zone for a DHCP pool like this:

```
$ORIGIN 2.0.192.in-addr.arpa.
$GENERATE 1-254 $ PTR dhcp-$.example.com.
```


This expands to:

```
$ORIGIN 2.0.192.in-addr.arpa.
1                PTR        dhcp-1.example.com.
2                PTR        dhcp-2.example.com.
; ... 250 more records ...
253              PTR        dhcp-253.example.com.
254              PTR        dhcp-254.example.com.
```

3. Motivation

One of the problems encountered when assigning address space with a longer prefix (fewer addresses) is that it seems impossible for the user of the address space to maintain their own reverse ("IN-ADDR.ARPA") zone autonomously. This obstacle can be overcome using the reverse delegation method described below.

Let us assume we have assigned the address spaces to three different parties as follows:

```
192.0.2.0/25   to organization A
```

```
192.0.2.128/26 to organization B
192.0.2.192/26 to organization C
```

In the classical approach, this would lead to a single zone like this:

```
$ORIGIN 2.0.192.in-addr.arpa.
;
1               PTR     host1.A.example.
2               PTR     host2.A.example.
3               PTR     host3.A.example.
;
129             PTR     host1.B.example.
130             PTR     host2.B.example.
131             PTR     host3.B.example.
;
193             PTR     host1.C.example.
194             PTR     host2.C.example.
195             PTR     host3.C.example.
```

The administration of this zone is problematic. Authority for this zone can only be delegated once, and this usually translates into "this zone can only be administered by one organization." The other organizations with address space that corresponds to entries in this zone would thus have to depend on another organization for their address to name translation. This potential problem can be avoided

using the method described in this memo.

4. Classless IN-ADDR.ARPA delegation for long prefixes

This section describes classless IN-ADDR.ARPA delegation for prefix lengths between /25 and /31 inclusive.

Since a single zone (such as 2.0.192.in-addr.arpa) can only be delegated once, we need more delegation points to solve our problem. An extra delegation point can be introduced by extending the IN-ADDR.ARPA tree downwards, by adding a label that is not entirely numeric so it does not clash with the existing reverse DNS names.

For each /24 subdivided up using this method, there are 256 CNAME records in the parent zone pointing into the child zones via these extra delegation points. It is quite easy to automatically generate the CNAME resource records in the parent zone once and for all, after you know the way the address space is partitioned.

Continuing the motivating example given in Section 3, here is how you can divide a /24 into a /25 and two /26 ranges.

```
$ORIGIN 2.0.192.in-addr.arpa.
@       IN      SOA     ns0.isp.example. ( ... )
; ...
;
0-127           NS      ns1.A.example.
0-127           NS      ns2.A.example.
;
$GENERATE 0-127 $ CNAME $.0-127
;
128-191         NS      ns1.B.example.
128-191         NS      ns2.B.example.
;
$GENERATE 128-191 $ CNAME $.128-191
;
192-255         NS      ns1.C.example.
192-255         NS      ns2.C.example.
;
$GENERATE 192-255 $ CNAME $.192-255
```

In this example the extra delegation points are named after the bottom and top addresses in the delegated address range, using the same format as the $GENERATE range specifier.

The $GENERATE directives produce 256 CNAME records which when expanded look like:

```
42.2.0.192.in-addr.arpa.  CNAME  42.0-127.2.0.192.in-addr.arpa.
```

The child zones might look something like:

```
$ORIGIN 0-127.2.0.192.in-addr.arpa.
```

```
@       IN      SOA     ns0.A.example. ( ... )
                NS      ns1.A.example.
                NS      ns2.A.example.
;
1               PTR     host1.A.example.
2               PTR     host2.A.example.
3               PTR     host3.A.example.


$ORIGIN 128-191.2.0.192.in-addr.arpa.
@       IN      SOA     ns0.B.example. ( ... )
                NS      ns1.B.example.
                NS      ns2.B.example.
;
129             PTR     host1.B.example.
130             PTR     host2.B.example.
131             PTR     host3.B.example.


$ORIGIN 192-255.2.0.192.in-addr.arpa.
@       IN      SOA     ns0.C.example. ( ... )
                NS      ns1.C.example.
                NS      ns2.C.example.
;
193             PTR     host1.C.example.
194             PTR     host2.C.example.
195             PTR     host3.C.example.
```

When a client does a reverse DNS query for an IP address in this range, it will get an answer like this:

```
;; QUESTION SECTION:
;42.2.0.192.in-addr.arpa.   IN   PTR

;; ANSWER SECTION:
42.2.0.192.in-addr.arpa.  CNAME  42.0-127.2.0.192.in-addr.arpa.
42.0-127.2.0.192.in-addr.arpa.  PTR  host42.A.example.
```

(TTL and CLASS omitted to save space.)

5. IN-ADDR.ARPA delegation for individual addresses

This section describes IN-ADDR.ARPA delegation for prefix lengths of /32, that is, individual IP addresses.

The CNAME trick described in the previous section is not necessary when delegating the reverse DNS for an individual IP address. Instead you can delegate at the reverse DNS name itself (for example, delegate at 42.2.0.192.in-addr.arpa), and put the PTR record at the apex of the delegated zone.

In detail (not continuing the previous examples), say isp.example has delegated the reverse DNS for 192.0.2.42/32 to their customer K.example:

```
$ORIGIN 2.0.192.in-addr.arpa.
@       IN      SOA     ns0.isp.example. ( ... )
; ...
42              NS      ns1.K.example.
42              NS      ns2.K.example.
; ...
```

The delegated zone will only have a few records at its apex:

```
$ORIGIN 42.2.0.192.in-addr.arpa.
@       IN      SOA     ns0.K.example. ( ... )
                NS      ns1.K.example.
                NS      ns2.K.example.
                PTR     host42.K.example.
```

The CNAME method described in Section 4 is in fact not necessary. You can delegate the reverse DNS for a CIDR address block by setting up delegations and /32 zones for every address in the block. However it is usually simpler to set up a single delegated zone and an automatically-generated set of CNAME records.

6. Classless IN-ADDR.ARPA delegation for short prefixes

This section describes classless IN-ADDR.ARPA delegation for prefix lengths between /9 and /23 inclusive, except for /16 which falls on an octet boundary.

Just as you can replace lots of /32 zones with one CIDR zone and some CNAME records, you can replace lots of /24 or /16 zones with one CIDR zone and some DNAME records. It is equally easy to set up, though the way it works is a bit more complicated.

DNAME records are described in [RFC6672], but here is a very terse summary. Whereas a CNAME record acts as a redirect for its owner name, a DNAME record acts as a redirect for descendents of its owner name, but not the name itself. Whereas a wildcard CNAME redirects a subtree of the DNS namespace to a single target name, a DNAME redirects a subtree to corresponding names in a different subtree.

Say, for example, that two organizations A and B want to share 10.0.0.0/8, one using the bottom half and the other using the top half. But they would each prefer not to have to manage 128 master zones of their own and 128 secondary zones from their counterpart.

They can reduce this from 256 zones (one for each /16) to 2 zones (one for each /9) by setting up the parent zone like this:

```
$ORIGIN 10.in-addr.arpa.
@       IN      SOA     ns0.B.example. ( ... )
; ...
;
0-127           NS      ns1.A.example.
0-127           NS      ns2.A.example.
;
$GENERATE 0-127 $ DNAME $.0-127
;
128-255         NS      ns1.B.example.
128-255         NS      ns2.B.example.
;
$GENERATE 128-255 $ DNAME $.128-255
```

The $GENERATE directives produce 256 DNAME records which when expanded look like:

```
2.10.in-addr.arpa.  DNAME  2.0-127.10.in-addr.arpa.
```

This DNAME record has the effect of mapping names under 2.10.in-addr.arpa to corresponding names under 2.0-127.10.in-addr.arpa, like this:

```
4.3.2.10.in-addr.arpa  ->  4.3.2.0-127.10.in-addr.arpa
```

The child zones will look something like,

```
$ORIGIN 0-127.10.in-addr.arpa.
@       IN      SOA     ns0.A.example. ( ... )
                NS      ns1.A.example.
                NS      ns2.A.example.
```

```
;
4.3.2           PTR     host4.A.example. ; 10.2.3.4
```



```
$ORIGIN 128-255.10.in-addr.arpa.
@       IN      SOA     ns0.B.example. ( ... )
                NS      ns1.B.example.
                NS      ns2.B.example.
;
20.100.200      PTR     host20.B.example. ; 10.200.100.20
```

When a client does a reverse DNS query for an IP address in this range, it will get an answer containing a DNAME record, a synthesized CNAME record, and the canonical answer:

```
;; QUESTION SECTION:
;4.3.2.10.in-addr.arpa.  IN  PTR

;; ANSWER SECTION:
2.10.in-addr.arpa.  DNAME  2.0-127.10.in-addr.arpa.
4.3.2.10.in-addr.arpa.  CNAME  4.3.2.0-127.10.in-addr.arpa.
4.3.2.0-127.10.in-addr.arpa.  PTR  host4.A.example.
```

(TTL and CLASS omitted to save space.)

7. Alternative naming conventions

In the sections above, we have suggested naming CIDR subdomains using the same notation as $GENERATE ranges, first-last. However the choice of name is just convention, and you are free to use a different convention if that is more convenient for you.

7.1. Subnet base slash prefix length (a bad idea)

In its main example, the predecessor to this memo [RFC2317] suggested using the subnet's first address and prefix length separated by a slash, for instance,

```
$ORIGIN 2.0.192.in-addr.arpa.
129     CNAME   129.128/26.2.0.192.in-addr.arpa.
```

One problem with putting a slash in the zone name is that it is common to give zone master files the same name as the zone they describe; this is troublesome if the zone name includes a directory separator.

In fact, [RFC2317] went on to say this convention is a bad idea. The remainder of this section quotes the paragraphs that explain why you should not follow this example:

Some DNS implementations are not kind to special characters in domain

names, e.g. the "/" used in the above examples. As [RFC2181] makes clear, these are legal, though some might feel unsightly. Because these are not host names the restriction of [RFC0952] does not apply. Modern clients and servers have an option to act in the liberal and correct fashion.

The examples here use "/" because it was felt to be more visible and pedantic reviewers felt that the 'these are not hostnames' argument needed to be repeated. We advise you not to be so pedantic, and to not precisely copy the above examples, e.g. substitute a more conservative character, such as hyphen, for "/".

7.2. Subnet base hyphen prefix length

The sensible version of the [RFC2317] convention is to use the subnet's first address and prefix length separated by a hyphen, for instance,

```
$ORIGIN 2.0.192.in-addr.arpa.
129     CNAME   129.128-26.2.0.192.in-addr.arpa.
```

This convention has the advantage of matching the way that address assignments are configured for routing. However the first-last convention is able to use a single DNS delegation covering adjacent CIDR blocks; for instance, a /25 and an adjacent /26 can be covered by a single delegation for 0-191.

7.3. Subnet base only (a bad idea)

Another alternative mentioned in [RFC2317] is the subnet's first address by itself.

With this convention, most addresses in the subnet have a CNAME (as in Section 4), but the subnet base address does not and instead looks like a /32 delegation (as in Section 5).

Because of its lack of uniformity we discourage you from using this convention.

7.4. Customer ID

It is not necessary for the subdomain to be tied to an address range; it could instead be a customer name or other ID. Then if that customer's address allocation changes, their provider can just add or remove CNAME or DNAME records without having to change the delegation.

This convention can also be useful if two organizations somehow share

the same physical subnet (and corresponding IP address space) with no "neat" split between the allocations, but they still want to administrate their own IN-ADDR.ARPA mappings.

7.5. In the forward DNS tree

The CNAME or DNAME records do not have to point into a subdomain: it is also possible to point to an entirely different part of the DNS tree, that is, outside of the IN-ADDR.ARPA tree. This variant of our running example might look like,

```
$ORIGIN 2.0.192.in-addr.arpa.
@        IN      SOA     ns0.isp.example. ( ... )
; ...
$GENERATE 0-127   $ CNAME $.A.example.
$GENERATE 128-191 $ CNAME $.B.example.
$GENERATE 192-255 $ CNAME $.C.example.

$ORIGIN A.example.
@        IN      SOA     ns0.A.example. ( ... )
; ...
;
host1            A       192.0.2.1
1                PTR     host1
;
```

```
host2           A       192.0.2.2
2               PTR     host2
;
; ...
```

This way you can actually end up with the name -> address and the (pointed-to) address -> name mapping data in the same zone file. The two mutually inverse mappings can be updated in the same DNS UPDATE transaction (though this benefit should not be exaggerated: the records will be still be cached separately and will time out independently). Do however note that the resolver's traversal via the IN-ADDR.ARPA tree will still be done, so the CNAME records inserted there need to point to the right place for this to work.

8. Consolidated reverse DNS zones

This section describes how to reduce the number of reverse DNS delegations. It applies to any prefix length, and can be used for IPv6 as well as IPv4.

Section 7.5 suggests setting up DNAME and/or CNAME records to point from the reverse DNS tree to the forward DNS. A significant

advantage of this is that it eliminates a delegation: instead of having to agree on a CIDR subdomain name, an NS RRset, and a DS RRset, and keep these up-to-date as name servers and DNSSEC keys change, you only need to agree on a target consolidation domain name.

The convention we recommend is to lay out part of your forward DNS namespace in the same way as the standard reverse DNS. That is, set up an "in-addr" subdomain under which you place PTR records for reversed dotted-quad IPv4 addresses, and/or an "ip6" subdomain under which you place PTR records for reversed exploded IPv6 addresses. This allows you to consolidate the reverse DNS for multiple disparate address blocks into the same zone. (You can have separate consolidation zones for IPv4 and IPv6, and for public and private addresses.)

For example, say organization A decides to consolidate their reverse zones. They would set up their forward DNS like this:

```
$ORIGIN A.example.
@               IN      SOA     ns0.A.example. ( ... )
; ...
;
1.2.0.192.in-addr       PTR     host1
2.2.0.192.in-addr       PTR     host2
4.3.2.10.in-addr        PTR     host4
```

For long prefixes like 192.0.2.0/25, organization A asks their provider to point CNAME records at their consolidation domain under A.example, like this:

```
$ORIGIN 2.0.192.in-addr.arpa.
; ...
$GENERATE 0-127 $ CNAME $.2.0.192.in-addr.A.example.
```

For short prefixes like 10.0.0.0/9, organization A asks their provider to point DNAME records at their consolidation domain under A.example, like this:

```
$ORIGIN 10.in-addr.arpa.
; ...
$GENERATE 0-127 $ DNAME $.10.in-addr.A.example.
```

Similarly, for its IPv6 network 2001:db8:A::/48, organization A again asks for a DNAME record, like this:

```
$ORIGIN 8.b.d.0.1.0.0.2.ip6.arpa.
; ...
A.0.0.0   DNAME   A.0.0.0.8.b.d.0.1.0.0.2.ip6.A.example.
```


To get the full benefit of a consolidated reverse zone, DNAME records should be used instead of delegations. However this requires co-operation from the provider of the address space.

Instead of inserting DNAME records in the provider's reverse DNS zone, you can add delegations on octet boundaries as usual, and put DNAME records at the apex of the delegated zones. (Unlike CNAMEs, DNAMEs do not conflict with other records at the same name.) This makes the reverse zones small and static, which is a small advantage, though it does not avoid the other overheads of managing a delegation.

9. Dynamic DNS UPDATE for reverse DNS pointers

This section updates the DNS UPDATE specification [RFC2136]. It specifies additional requirements for DNS UPDATE clients, so they can dynamically change reverse DNS records in a way that is compatible with the techniques described in the previous sections. It applies both to the IPv4 reverse DNS under IN-ADDR.ARPA and the IPv6 reverse DNS under IP6.ARPA.

These additional requirements only apply to DNS UPDATE clients that wish to add, remove, or change endpoint records in the reverse DNS. These requirements do not apply if you are using DNS UPDATE for other purposes, such as altering zone apex or delegation records, or CNAME or DNAME records - which you need to do if you are using DNS UPDATE to deploy classless IN-ADDR.ARPA delegations. These requirements do not affect uses of DNS UPDATE outside the IN-ADDR.ARPA and IP6.ARPA sub-trees.

In this section, we use the term "reverse DNS query name" to mean a name under IN-ADDR.ARPA or IP6.ARPA which a resolver uses when making a reverse DNS query. The resolver expects this name to resolve to a PTR RRset or other endpoint records, but (as described in previous sections) the query name does not have to be the direct owner of the endpoint records but can instead be an alias.

We use the term "endpoint records" as a generalization of the PTR RRset, since the reverse DNS can include other information about an IP address - not just its host name. For instance, the EUI48 and EUI64 RRtypes are intended for mapping from IP addresses to MAC addresses [RFC7043].

The problem addressed by this section is that DNS UPDATE clients sometimes use a reverse DNS query name in an UPDATE message without checking for CNAME or DNAME redirections. If the usual reverse DNS query name is an alias, then this behavior results in an attempt to

add or delete an endpoint record to or from a node that already contains the CNAME record, and the update fails.

Aside: Presumably the UPDATE will also fail if the node is occluded below a DNAME record, but neither [RFC2136] nor [RFC6672] specifies how a server ought to react to attempts to UPDATE an occluded domain name.

9.1. Requirements for updating PTR records in the reverse DNS

When updating a endpoint records in the reverse DNS, an UPDATE client SHOULD NOT simply convert the IP address to a reverse DNS query name and send an UPDATE request for the records at that name. It MUST NOT assume the zone cut falls on a particular boundary such as /24 for IPv4 or /64 for IPv6.

Instead, the UPDATE client SHOULD canonicalize all reverse DNS query names that it uses in its UPDATE message. It MUST ensure that all the canonical names are within the same zone and that the ZNAME field in the UPDATE message refers to this zone.

9.2. Suggested behaviour

In the absence of more specific configuration, a reverse DNS UPDATE client can follow this procedure.

Send a SOA query for the reverse DNS query name. There are four kinds of useful response.

- o There is a SOA record at the query name, which is returned in the answer section of the response. This can occur if the name has a /32 delegation.

- o The query name is an alias for a name with a SOA record. The answer section of the response contains a CNAME chain and a SOA record.

- o The query name is not an alias. The response is NOERROR or NXDOMAIN. The answer section is empty, and the authority record contains a SOA record. This is the traditionally expected result.

- o The query name is an alias. The response is NOERROR or NXDOMAIN. The answer section has a CNAME chain, and the authority record contains a SOA record. This is the normal case for classless IN-ADDR.ARPA delegations or consolidated reverse DNS.

In other cases there has been some kind of problem and the DNS UPDATE cannot proceed.

(Note that if the reverse DNS query name is under a DNAME, the response will contain a DNAME in the answer section and a synthesized CNAME. The UPDATE client can ignore the DNAME and just use the CNAME records.)

(There can also be other records in the response, but they are not relevant to this procedure and so are not discussed here.)

All of the useful responses give the DNS UPDATE client enough information to construct a correct UPDATE message.

- o The server to send the UPDATE message to comes from the SOA MNAME field.
- o The UPDATE ZNAME comes from the owner name of the SOA record.
- o The canonical name of the endpoint records comes from the final target of the CNAME chain, or the QNAME if there are no CNAME records in the answer.

10. Operational considerations

10.1. Secondary name service

Very old name server software might not find and return the target name in CNAME records if the targte name is not already known locally as cached or as authoritative data. This can cause some confusion in stub resolvers, as only the CNAME record will be returned in the response. To avoid this problem it is recommended that the authoritative name servers for the delegating zone (the zone containing all the CNAME or DNAME records) all run as secondary name servers for the target zones delegated and pointed into via the CNAME/DNAME records.

10.2. CNAME chains

Multiple levels of delegation using the methods described in this memo lead to multi-step CNAME and/or DNAME chains. Although [RFC1034] requires resolvers to handle CNAME chains robustly, such a setup might be less reliable overall.

10.3. Mail servers

SMTP servers are often very picky about reverse DNS, and some are known to be intolerant of DNAME records. Therefore it is wise to be wary of deploying the methods described in Section 6 and Section 8 for IP address ranges that contain outgoing inter-domain mail

senders.

10.4. Workaround for DNS UPDATE interoperability problems

If you have the problem described in Section 9 because your reverse DNS UPDATE client does not follow the new requirements in that section, you might be able to work around it by using /32 delegations as described in Section 5. This allows you to eliminate the CNAME records used by the normal classless delegation method (Section 4) at the cost of requiring more zones. And with /32 delegations, the canonical owner name of the PTR records is the usual reverse DNS query name, so the UPDATE client does not need to chase CNAME chains.

11. Security Considerations

11.1. Classless delegation

With this scheme, the "leaf sites" might need to rely on one more site running their DNS name service correctly than they would be if they had a /24 allocation of their own, and this might add an extra component which will need to work for reliable name resolution.

11.2. Consolidated reverse zones

Normal reverse DNS delegations and classless delegations require more frequent changes when zones are signed for DNSSEC [RFC4033], to update the DS records in the parent zone to track key rollovers. Consolidated reverse zones (Section 8) replace delegations with CNAME and/or DNAME pointers. This reduces the number of secure delegations that must be managed, which should make operations simpler and more robust; however it means that reverse DNS resolution depends on chains of trust in the forward DNS as well as the reverse DNS. This does not necessarily increase the number of trusted entities in a meaningful way, if the consolidated reverse zone is in the same part of the namespace as the targets of the PTR records.

11.3. Reverse DNS UPDATE

The canonicalization process changes the owner name that will be affected by the update. An active attacker might interfere with the canonicalization process and trick the requestor to update a node of the attacker's choice if the canonicalization process is not secured by using TSIG [RFC2845] or DNSSEC [RFC4033] or other means.

When using DNSSEC, an implementation might decide to accept canonicalized names only on condition that the overall security status of the canonicalization process is sufficient according to the

local policy. Because the chain of redirections might involve multiple DNS zones, implementations MUST use the lowest security status from all links in the chain of redirections when doing security decisions.

12. Acknowledgments

Glen A. Herrmannsfeldt described this technique (specifically the Section 7.4 variant) on the comp.protocols.tcp-ip.domains newsgroup in 1991 [GAH1] and in more detail in 1994 [GAH4]. Alan Barrett and Sam Wilson provided valuable comments.

Chris Thompson described the technique in Section 8 on the bind-users mailing list in 2009 [CET1] [CET2]. Chris Hills and Niall O'Reilly confirmed they had also deployed it.

Petr Spacek pointed out the need to clarify the behaviour of dynamic DNS UPDATE requests for reverse DNS mappings [I-D.spacek-dnsop-update-clarif].

Thanks to the authors of [RFC2317]: Havard Eidnes, Geert Jan de Groot, and Paul Vixie.

Thanks to the following people for their helpful comments about this memo: Peter van Dijk, Bob Harold, Niall O'Reilly, Petr Spacek, Chris Thompson, Paul Vixie.

13. References

13.1. 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>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)",

RFC 2136, DOI 10.17487/RFC2136, April 1997, <http://www.rfc-editor.org/info/rfc2136>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <http://www.rfc-editor.org/info/rfc6672>.

13.2. Informative References

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <http://www.rfc-editor.org/info/rfc952>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <http://www.rfc-editor.org/info/rfc2181>.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <http://www.rfc-editor.org/info/rfc2317>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <http://www.rfc-editor.org/info/rfc2845>.

[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>.

[RFC5156] Blanchet, M., "Special-Use IPv6 Addresses", RFC 5156, DOI 10.17487/RFC5156, April 2008, <http://www.rfc-editor.org/info/rfc5156>.

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", RFC 5735, DOI 10.17487/RFC5735, January 2010, <http://www.rfc-editor.org/info/rfc5735>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <http://www.rfc-editor.org/info/rfc6761>.

[RFC7043] Abley, J., "Resource Records for EUI-48 and EUI-64 Addresses in the DNS", RFC 7043, DOI 10.17487/RFC7043, October 2013, <http://www.rfc-editor.org/info/rfc7043>.

[RFC7535] Abley, J., Dickson, B., Kumari, W., and G. Michaelson, "AS112 Redirection Using DNAME", RFC 7535, DOI 10.17487/RFC7535, May 2015, <http://www.rfc-editor.org/info/rfc7535>.

[I-D.spacek-dnsop-update-clarif] Spacek, P., "Clarifications to the Dynamic Updates in the Domain Name System (DNS UPDATE) specification", draft-spacek-dnsop-update-clarif-01 (work in progress), August 2015.

[GAH1] Herrmannsfeldt, G., "Re: Subnets and IN-ADDR.ARPA", April 1991, <https://groups.google.com/d/original/comp.protocols.tcp-ip.domains/39RWpZ549AY/T10zM45oT88J>.

[GAH4] Herrmannsfeldt, G., "Re: Nonstandard subnetting", December 1994, <https://groups.google.com/d/original/comp.protocols.tcp-ip.domains/ejo4z2CTVLc/s3sTyhjDtOQJ>.

[CET1] Thompson, C., "Pruning the reverse zone tree", February 2009, <https://lists.isc.org/pipermail/bind-users/2009-February/075240.html>.

(Message to the bind-users mailing list)

[CET2] Thompson, C., "Pruning the reverse zone tree", February 2009, <https://jackdaw.cam.ac.uk/ipreg/nsconfig/prune-reverse-zones.html>.

(Maintained copy of the prune-reverse-zone notes)

Appendix A. Changes since RFC2317

- Use a recommended naming convention in the main example. Clearly describe which alternative conventions are good and bad ideas.
- Add a description of /32 delegations.
- Add a description of using DNAME for delegations on short prefixes.
- Emphasize the consolidated reverse DNS convention.

- Add a description of $GENERATE and use it in the examples.
- Specify new requirements on dynamic reverse DNS UPDATE clients.
- Better citations for Glen Herrmannsfeldt's original description.

Appendix B. Questions for reviewers

Is the $GENERATE section a good idea? Should it be ditched? Or maybe promoted to its own document?

RFC 2317 is BCP 20. Should this document be moved to the standards track, since it updates RFC 2136? Or should the UPDATE amendment be a separate document?

I have generally avoided RFC 2119 keywords in the sections describing how to set up classless delegations, since those sections contain operational advice rather than implementation requirements. Other opinions welcome.

Is the indirection problem specific to classless reverse DNS (which is the approach I took) or does it apply to the forward DNS as well? Suggestions for wording welcome.

Is the detailed UPDATE behaviour sensible?

Appendix C. Changelog

Note to RFC editor: This section should be removed before publication. The important points should appear in the previous section.

A detailed revision log can be found at <https://git.csx.cam.ac.uk/x/ucs/u/fanf2/rfc2317bis.git>.

C.1. Chnages between -00 and -01

- o Note troublesome zone file names.
- o Clarify DNAME-at-apex
- o Notes on first-last vs first-prefixlen
- o Compatibility note about $GENERATE

- o Note possible workaround for reverse DNS UPDATE canonicalization problem.
- o Accommodate more than just PTR records in the reverse DNS.

o Incorporate some security considerations from [I-D.spacek-dnsop-update-clarif]

Author's Address

Tony Finch
University of Cambridge Information Services
Roger Needham Building
7 JJ Thomson Avenue
Cambridge CB3 0RB
England

Phone: +44 797 040 1426
Email: dot@dotat.at