Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 17, 2012

J. Gersch
Secure64 SW Corp
D. Massey
Colorado State University

February 14, 2012

Reverse DNS Naming Convention for CIDR Address Blocks
draft-gersch-dnsop-revdns-cidr-00.txt

Abstract

The current reverse DNS naming method is used to specify a complete IP address. It has not been used to handle address ranges; for example, there is no formal mechanism for specifying a reverse DNS name for the block of addresses specified by the IPv4 prefix 129.82.0.0/16. Defining such a reverse DNS naming convention would be useful for a number of applications. These include applications for secure BGP routing, and applications that need host-information for a device owning a complete IPv6 address block. This draft proposes a naming convention for encoding CIDR address blocks in the reverse DNS.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This draft proposes a common naming convention for entering CIDR prefixes into the Reverse DNS.

The Reverse DNS provides a naming convention for both IPv4 and IPv6 addresses. At this time, the primary use of the reverse-DNS is to associate an IP address with a PTR resource record that identifies the corresponding host name. For example, IP address 129.82.138.2 is encoded as 2.138.82.129.in-addr.arpa and a PTR resource record identifies the host name as alpha.netsec.colostate.edu. The Reverse DNS would become more powerful if it could also return information associated with a network address range, not just a unique IP address. For example, one would like to store and resolve records associated with a prefix range such as 129.82.128/17.

Given such a capability, a variety of new applications and services would be enabled. For example, internet routing operators could publish authorized BGP route origins for their network address blocks in the reverse-DNS. Another application could query for a set of host-names or services associated with an address block; for example, to indicate the authorized mail servers for an address block.

Yet another interesting possibility is to solve a problem with IPv6 dynamic DNS assignments. In IPv4, the owner of address block could simply include one PTR record for every available address. In fact, ISPs commonly pre-populate the reverse DNS zone for their customers. However, this approach clearly does not scale for IPv6 where the number of addresses becomes excessively large. For example, allocation of a /48 (not uncommon in IPv6) includes 2^80 addresses and notes adding 1000 PTR records per second would require over 38 trillion years to pre-populate the reverse DNS [I-D.howard-isp-ip6rdns]. The ability to name prefix blocks rather than individual addresses could help address this problem by publishing records associated with an entire IPv6 address range instead of replicating or synthesizing answers to unique address queries.

The above list of possible applications is not intended to be complete, but instead suggest some of the possibilities.

1.1. Purpose

In order to enable these applications, one must map an IPv4 or IPv6 prefix into a reverse-DNS name. There are various subtleties, advantages and disadvantages that emerge when trying to define a naming convention. Today, zone administrators can use their own individual approaches to encode a prefix in the reverse DNS. This

requires no DNS protocol changes and no modifications to resolvers, caches, or authoritative servers. The emergence of different encoding standards complicates (but does not prevent) the design of systems that would make use of these resource records. The aim of this work is to introduce a standard convention.

1.2. Terminology

The following terms are used througt out the document:

Reverse DNS:
We use the term Reverse DNS to refer to the domains in-addr.arpa and ip6.arpa.

Prefix:
A prefix refers to IPv4 or IPv6 address range specified by a network portion and mask length, as described in [RFC4632]. For example, 129.82.0.0/16 and 129.82.128/18 are examples of IPv4 prefixes.

Octet Boundary:
An IPv4 prefix falls on an octet boundary if its mask length is a multiple 8. For example, 129.82.0.0/16 is on an octet boundary while 129.82.128/18 does not fall on octet boundary. Prefixes that are on octet boundary naturally map to the reverse DNS. Prefixes that are not on octet boundary are more complex and the main challenge for any naming convention.

2. 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].

3. Design Requirements

A naming convention to specify CIDR address blocks in the reverse-DNS has several design goals:

1. Autonomy: The owner of a reverse-DNS zone file associated with a CIDR address block must be able to act independently from any other organization in order to create or modify data records within the DNS zone.

2. Coverage Authority: With the exception of data that has been sub-delegated to a child zone, the reverse DNS zone must be authoritative for all sub-prefixes below the covering prefix. Any query for a sub-prefix must be answered with a data record or NXDOMAIN specifying this zone as the authority.

3. Allow Delegation: It must allow the zone owner to delegate smaller address blocks to a child zone which will be independently managed.

4. Conformance: It should align with naming conventions and delegation structures already in use by the RIR's for IN-ADDR.ARPA and IP6.ARPA.

5. Simplicity: The naming structure should be understandable, or at a minimum, able to be easily constructed by software provisioning tools and utilities such as DIG.

4. Related Work

The process of mapping CIDR addresses into the reverse-DNS name space is difficult because the prefix length of an IPv4 CIDR address is an arbitrary number from 0 to 32. These numbers do not necessarily align with an IPv4 octet.

4.1. CIDR Naming via RFC 2317

Since CIDR address no longer align with octet boundaries, the CIDR specification in [RFC4632] notes that there is "some increase in work for those who maintain parts of the IN-ADDR.ARPA zone." [RFC2317] is offered a technique to populate the IN-ADDR.ARPA. The intent of this work is to encode IPv4 addresses and the approach is designed to "address spaces covering fewer than 256 addresses."

Suppose organization A owns 129.82.138.0/30. This address space covers four IPv4 addresses; namely 129.82.138.0, 129.82.138.1, 129.82.138.2 and 129.82.138.3. Giving organization A control of the reverse zone "138.82.129.in-addr.arpa." would allow Organization A to enter PTR resource records for each of its 4 addresses. However, it also gives organization A the ability to enter PTR resource records for 252 other IP addresses from 129.82.138.4 to 129.82.138.255. These addresses are managed by other organizations. Sharing the 138.82.129.in-addr.arpa between multiple organization is not practical and creating a seperate zone for each IP address (e.g. creating the zone 0.138.82.129.in-addr.arpa) is very high overhead to store a single PTR record.

[RFC2317] addresses this problem by creating CNAME records in 138.82.129.in-addr.arpa zone. Organization A administers a zone named 0/32.138.129.in-addr.arpa. CNAME records in the 138.82.129.in-addr.arpa zone point to entries in Organization A's 0/32.138.82.129.in-addr.arpa zone. For example, 1.138.82.129.in-addr.arpa. is a CNAME pointing to 1.0/32.138.82.129.in-addr.arpa. A full description is found in [RFC2317].

This approach was not intended to encode IP address for address spaces smaller than a "/24". It was not intended for encoding prefixes. It does not specify how one might encode a prefix and it is not trivial to extend this approach to CIDR prefixes. In particular, the design requirements of Coverage Authority, Allowing Delegation, and arguably Simplicity are not easily met by extending the RFC to included prefixes.

4.2. Prior Work on CIDR Names for Routing

Over a decade ago, [I-D.bates-bgp4-nlri-orig-verif] proposed to use the reverse DNS to verify the origin AS associated with a prefix. This requires both a naming convention for converting the name into a prefix and additional resource record types for storing origin information, along with recommendations on their use.

Our focus in this draft is on the naming convention. The draft extends [RFC2317] style names to encode a prefix. For example, the draft proposes to encode the prefix 10.1.128/20 as the DNS name 128/20.1.10.bgp.in-addr.arpa. This fails to meet the Coverage Authority requirement. To see this, consider a more specific prefix such as 10.1.128/21. 10.1.128/21 is encoded as 128/21.1.10.bgp.in-addr.arpa.

The problem is that in CIDR terminology, 10.1.128/21 is covered by 10.1.128/20. But in the DNS structure, 10.1.128/20 and 10.1.128/21 are siblings in the DNS tree structure. This can be overcome by introducing a large number of CNAME records (one for every potential subprefix), but we seek an approach where the CIDR structure and DNS hierarchy align.

5. Reverse DNS CIDR Name Specification

The naming method described in this section is based on the well-known technique of ANDing a bit-mask with the low-order octet of an IP address. The binary result is then broken up into individual sub-names using the "." separator. The result looks like an ENUM or IPv6 reverse-DNS address; that is, a string of chained empty non-terminal sub-names.

This name-chaining creates the desired effect of being able to allow a DNS zone delegation at any point in the chain. The naming scheme allows the creation of two /17's from a /16, two /18's from a /17, and so on.

5.1. IPv4 Address Block Naming

The CIDR to Reverse-DNS naming convention works as follows:

1. Invert the address per the usual reverse-DNS method. Remove any trailing zeroes: 129.82.0.0/16 --> 82.129.in-addr.arpa.

2. Calculate N where N = prefix-length mod 8. Stop If N equals 0. The name conversion is complete because you are at an octet boundary. Otherwise:

3. Perform the following name construction:

   A. Truncate the original name to remove the least significant non-zero octet. Add ".m" characters to this string to indicate "mask".

   B. Convert the least significant octet to binary, separating each digit with a "." character.

   C. Truncate the binary digits to the N significant binary characters that correspond to the given prefix-length.

   D. Reverse the string and add ".in-addr.arpa."

Several examples will illustrate this algorithm. These examples show the conversion to binary, followed by the truncation, followed by the name reversal.

```
129.82.0.0/16   --> 82.129.in-addr.arpa. (at octet boundary)


129.82.64.0/18  --> 129.82.m.0.1.0.0.0.0.0.0
                --> 129.82.m.0.1 (N = 18 mod 8 = 2)
```

```
                           --> 1.0.m.82.129.in-addr.arpa.


           129.82.64.0/20  --> 129.82.m.0.1.0.0.0.0.0.0
                           --> 129.82.m.0.1.0.0  (N = 20 mod 8 = 4)
                           --> 0.0.1.0.m.82.129.in-addr.arpa.


           129.82.160.0/20 --> 129.82.m.1.0.1.0.0.0.0.0
                           --> 129.82.m.1.0.1.0 (N = 20 mod 8 = 4)
                           --> 0.1.0.1.m.82.129.in-addr.arpa.


           129.82.160.0/23 --> 129.82.m.1.0.1.0.0.0.0.0
                           --> 129.82.m.1.0.1.0.0.0.0 (N = 23 mod 8 = 7)
                           --> 0.0.0.0.1.0.1.m.82.129.in-addr.arpa.


           15.192.0.0/12   --> 15.192.m.1.1.0.0.0.0.0.0
                           --> 15.192.m.1.1.0.0    (N = 12 mod 8 = 4)
                           --> 0.0.1.1.m.15.in-addr.arpa.
```

The conversion from a reverse-DNS name back to CIDR is simple. First calculate the prefix length from the name using the formula:

```
           plen = 8*(count of full octets) + (count of binary digits)
```

Then reverse the string, add up the values of the binary digits to build a final octet, then append a "/" and the prefix length.

Examples:

```
           1.0.m.82.129.in-addr.arpa  --> 129.82.64.0/18
           (example has 2 octets + 2 binary digits, so mask length = 18)


           0.0.1.0.m.82.129.in-addr.arpa --> 129.82.64.0/20
           (example has 2 octets + 4 binary digits, so mask length = 20)


           0.0.0.1.0.1.m.129.in-addr.arpa--> 129.160.0/14
           (example has 1 octet + 6 binary digits, so mask length = 14)
```

5.2. IPv4 Address Block Naming

The IPv6 naming convention is similar, with the exception that 4-bit nibble boundaries are used instead of octets, the mod calculation is based on 4 instead of 8, and "ip6.arpa" is used as the suffix.

Examples:

```
        2607:fa88::/32     --> 8.8.a.f.7.0.6.2.ip6.arpa
           (on nibble boundary)


        2607:fa88:8000:/33 --> 2.6.0.7.f.a.8.8.m.1.0.0.0
                           --> 2.6.0.7.d.a.8.8.m.1    (33 mod 4 = 1)
                           --> 1.m.8.8.a.f.7.0.6.2.ip6.arpa


        2607:fa88:e000:/35 --> 2.6.0.7.f.a.8.8.m.1.1.1.0
                           --> 2.6.0.7.d.a.8.8.m.1.1.1(35 mod 4 = 3)
                           --> 1.1.1.m.8.8.a.f.7.0.6.2.ip6.arpa
```

5.3. Special Case to Allow "Overlapping Names" at Octet Boundaries

In some instances it is desirable to create an "overlapping name" for a CIDR block located on an octet boundary. For example, rather than spread information in 256 separate /24 zone files, it would be more convenient to put all the 256 records in one parent zone. If an application is searching for data and does not find it in the /24 zone it may optionally decide to look in the parent zone to see if an overlapping name exists and use that data instead.

To construct an "overlapping name", use step 3 of the algorithm already described, skipping steps 1 and 2. N will always equal 8 for IPv4 and will equal 4 for IPv6. (Step 3C is not needed since there is nothing to truncate).

Examples:

```
        129.82.160.0/24 --> 129.82.m.1.0.1.0.0.0.0.0
                        --> 0.0.0.0.0.1.0.1.m.82.129.in-addr.arpa.


        129.82.255.0/24 --> 129.82.m.1.1.1.1.1.1.1.1
                  --> 1.1.1.1.1.1.1.1.m.82.129.in-addr.arpa.


        2607:fa88:e000:/36 --> 2.6.0.7.f.a.8.8.m.1.1.1.0
                           --> 0.1.1.1.m.8.8.a.f.7.0.6.2.ip6.arpa
```

6. Security Considerations

This document only introduces a naming convention. Applications that make use of this naming convention may require the use of DNSSEC to validate the resource records stored at these names.

7. IANA Considerations

This document does not request any IANA action.

8. Acknowledgments

This document was aided via numerous discussions at NANOG, IETF and private meetings with ISPs, telecomm carriers, and research organizations too numerous to mention by name. Thanks to all for your comments and advice.

9. References

9.1. Normative References

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

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

9.2. Informative References

[I-D.bates-bgp4-nlri-orig-verif]
Bates, T., Bush, R., Li, T., and Y. Rekhter, "DNS-based NLRI origin AS verification in BGP", draft-bates-bgp4-nlri-orig-verif-00 (work in progress), January 1998.

[I-D.howard-isp-ip6rdns]
Howard, L. and A. Durand, "Reverse DNS in IPv6 for Internet Service Providers", draft-howard-isp-ip6rdns-04 (work in progress), September 2010.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, March 1998.

Appendix A. Example Zone Files

A.1. Example 1

This example shows several DNS records added to an existing reverse-DNS zone file at octet boundary 129.82.0.0/16. The records show how BGP route origins for a CIDR prefix could be specified in the zone file. Otherwise no other changes were made.

Note: this internet draft is not proposing the RRTypes for routing shown here; they are only presented as sample content for the proposed naming convention.

```
     $TTL 3600
     $ORIGIN 82.129.in-addr.arpa.

     @    IN    SOA     rush.colostate.edu.  dnsadmin.colostate.edu. (
                             2012021300      ; serial number
                             900             ; refresh, 15 minutes
                             600             ; update retry, 10 minutes
                             86400           ; expiry, 1 day
                             3600            ; minimum, 1 hour
                            )

          IN    NS      dns1.colostate.edu.
          IN    NS      dns2.colostate.edu.

     @                  IN   TYPE3000 \# 0
     ;                  RLOCK    deny all route announcements
     ;                           except those authorized
     @                  IN   TYPE3001 \# 8 00002f71000036d9
     ; 129.82.0.0/16         route origin/nexthop AS12145 / AS14041
     0.0.m              IN   TYPE3001 \# 8 00002f71000036d9
     ; 129.82.0.0/18         route origin/nexthop AS12145 / AS14041
     1.0.m              IN   TYPE3001 \# 8 00002f71000036d9
     ; 129.82.64.0/18        route origin/nexthop AS12145 / AS14041
     0.1.m              IN   TYPE3001 \# 8 00002f71000036d9
     ; 129.82.128.0/18       route origin/nexthop AS12145 / AS14041
     1.1.m              IN   TYPE3001 \# 8 00002f71000036d9
     ; 129.82.192.0/18       route origin/nexthop AS12145 / AS14041

     1.0.0.0.1.1.0.1.m  IN   TYPE3001 \# 8 00004070000036d9
     ; 129.82.177.0/24       route origin/nexthop AS12145 / AS14041

     ;  delegations required for 256 /24 zones which contain PTR records

     1   IN  NS  dns1.colostate.edu.
         IN  NS  dns2.colostate.edu.
     2   IN  NS  dns1.colostate.edu.
         IN  NS  dns2.colostate.edu.

     ;  continuation to 255 is left out for the sake of brevity
```

In this first example we have added records with routing information pertinent to address blocks 129.82/16 and the four /18's at 129.82.0.0/18, 129.82.64.0/18, 129.82.128.0/18, and 129.82.192.0/18.

Finally, the example shows a record for a /24 using the full 8-bit overlapping notation so that the data can be placed in this parent

zone rather than in the child zone at 177.82.129.in-addr.arpa.

A.2. Example 2

This example illustrates the creation of a new zone for 216.17.128.0/17 which is not at an octet boundary.

The existing 256 zones delegated at IN-ADDR.ARPA for the range 0.17.128 through 255.17.216.in-addr.arpa remain unchanged; they contain PTR records maintained by the appropriate zone owners

Only a single new delegation needs to be added to IN-ADDR.ARPA:

```
        1.m.17.216.in-addr.arpa  NS   ns.frii.net
```

This delegation refers to the new /17 zone and is not in conflict with any of the pre-existing /24 zones.

```
   $TTL 3600
   $ORIGIN 1.m.17.216.in-addr.arpa.

   @    IN   SOA     ns1.frii.net.  hostmaster.frii.net. (
                           2012021300      ; serial number
                           14400           ; refresh, 4 hours
                           3600            ; update retry, 1 hour
                           604800          ; expiry, 7 days
                           600             ; minimum, 10 minutes
                          )

        IN   NS      ns1.frii.net.
        IN   NS      ns2.frii.net.

   $ORIGIN 17.216.in-addr.arpa.

   1.m               IN   TYPE3000 \# 0
   ;                 RLOCK    deny all route announcements
   ;                          except those authorized
   1.m               IN   TYPE3001 \# 8 000019b600000d1c
   ; 216.17.128.0/17      route origin/nexthop AS6582 / AS3356
   1.m               IN   TYPE3001 \# 8 000019b6000000ae
   ; 216.17.128.0/17      route origin/nexthop AS6582 / AS174

   ; no other delegations or PTR records are needed in this zone file
```

In this example we have added several records all at the same domain name with information pertinent to address block 216.17.128.0/17.

Authors' Addresses

Joe Gersch
Secure64 SW Corp
Fort Collins, CO
US

Email: joe.gersch@secure64.com

Dan Massey
Colorado State University
Fort Collins, CO
US

Email: massey@cs.colostate.edu