Internet Engineering Task Force
Internet-Draft

B. Halley
Nominum
E. Lewis
ARIN

June 17, 2003

Expires: December 17, 2003

Clarifying the Role of Wild Card Domains in the Domain Name System

<draft-ietf-dnsext-wcard-clarify-00.txt>

Status of this Memo

Abstract

The definition of wild cards is recast from the original in RFC 1034, in words that are more specific and in line with RFC 2119. This document is meant to supplement the definition in RFC 1034 and to alter neither the spirit nor intent of that definition.

1 Introduction

The first section of this document will give a crisp overview of what is begin defined, as well as the motivation for what amounts to a simple rewording of an original document. Examples are included to help orient the reader.

Wild card domain names are defined in Section 4.3.3. of RFC 1034 as "instructions for synthesizing RRs." [RFC1034] The meaning of this is that a specific, special domain name is used to construct responses in instances in which the query name is not otherwise represented in a zone.

A wild card domain name has a specific range of influence on query names (QNAMEs) within a given class, which is rooted at the domain name containing the wild card label, and is limited by explicit entries, zone cuts and empty non-terminal domains (see section 1.3 of this document).

Note that a wild card domain name has no special impact on the search for a query type (QTYPE). If a domain name is found that matches the QNAME (exact or a wild card) but the QTYPE is not found at that point, the proper response is that there is no data available. The search does not continue on to seek other wild cards that might match the QTYPE. To illustrate, a wild card owning an MX RR does not 'cover' other names in the zone that own an A RR.

Why is this document needed? Empirical evidence suggests that the words in RFC 1034 are not clear enough. There exist a number of implementations that have strayed (each differently) from that definition. There also exists a misconception of operators that the wild card can be used to add a specific RR type to all names, such as the MX RR example cited above. This document is also needed as input to efforts to extend DNS, such as the DNS Security Extensions [RFC 2535]. Lack of a clear base specification has proven to result in extension documents that have unpredictable consequences. (This is true in general, not just for DNS.)

Another reason this clarification is needed is to answer questions regarding authenticated denial of existence, a service introduced in the DNS Security Extensions [RFC 2535]. Prior to the work leading up to this document, it had been feared that a large number of proof records (NXTs) might be needed in each reply because of the unknown number of potential wild card domains that were thought to be applicable. One outcome of this fear is a now discontinued document solving a problem that is now known not to exist. I.e., this clarification has the impact of defending against unwarranted protocol surgery. It is not "yet another" effort to just rewrite the early specifications for the sake of purity.

1.1 Document Limits

This document limits itself to reinforcing the concepts in RFC 1034. Any deviation from this should be brought to the attention of the editors.

Two changes to the text of RFC 1034 that fall within the realm of clarifying the wild card definition have been suggested. (Changes aren't really clarifications.) The two suggestions are barring the ownership by a wild card domain of an CNAME resource record set and barring the ownership by a wild card domain of a NS resource record set. Both of these have some merit, but do not belong in a document that has not yet been reviewed by the working group.

1.2 Existence

The notion that a domain name 'exists' will arise numerous times in this

discussion. RFC 1034 raises the issue of existence in a number of places, usually in reference to non-existence and often in reference to processing involving wild card domain names. RFC 1034 does contain algorithms that describe how domain names impact the preparation of an answer and does define wild cards as a means of synthesizing answers.

To help clarify the topic of wild cards, a positive definition of existence is needed. Complicating matters, though, is the realization that existence is relative. To an authoritative server, a domain name exists if the domain name plays a role following the algorithms of preparing a response. To a resolver, a domain name exists if there is any data available corresponding to the name. The difference between the two is the synthesis of records according to a wild card.

For the purposes of this document, the point of view of an authoritative server is adopted. A domain name is said to exist if it plays a role in the execution of the algorithms in RFC 1034.

1.3 An Example

For example, consider this wild card domain name: *.example. Any query name under example. is a candidate to be matched (answered) by this wild card, i.e., to have an response returned that is synthesized from the wild card's RR sets. Although any name is a candidate, not all queries will match.

To further illustrate this, consider this example:

```
$ORIGIN example.
@        IN      SOA
                 NS
                 NS
*                TXT "this is a wild card"
                 MX  10 mailhost.example.
host1            A   10.0.0.1
_ssh._tcp.host1  SRV
_ssh._tcp.host2  SRV
subdel           NS
```

The following queries would be synthesized from the wild card:

```
QNAME=host3.example. QTYPE=MX, QCLASS=IN
      the answer will be a "host3.example. IN MX ..."
QNAME=host3.example. QTYPE=A, QCLASS=IN
      the answer will reflect "no error, but no data"
      because there is no A RR set at '*'
```

The following queries would not be synthesized from the wild card:

```
QNAME=host1.example., QTYPE=MX, QCLASS=IN
      because host1.example. exists
QNAME=_telnet._tcp.host1.example., QTYPE=SRV, QCLASS=IN
      because _tcp.host1.example. exists (without data)
```

```
QNAME=_telnet._tcp.host2.example., QTYPE=SRV, QCLASS=IN
      because host2.example. exists (without data)
QNAME=host.subdel.example., QTYPE=A, QCLASS=IN
      because subdel.example. exists and is a zone cut
```

To the server, the following domains are considered to exist in the zone: *, host1, _tcp.host1, _ssh._tcp.host1, host2, _tcp.host2, _ssh._tcp.host2, and subdel. To a resolver, many more domains appear to exist via the synthesis of the wild card.

1.4 Empty Non-terminals

Empty non-terminals are domain names that own no data but have subdomains. This is defined in section 3.1 of RFC 1034:

```
#    The domain name space is a tree structure.  Each node and leaf on the
#    tree corresponds to a resource set (which may be empty).  The domain
#    system makes no distinctions between the uses of the interior nodes and
#    leaves, and this memo uses the term "node" to refer to both.
```

The parenthesized "which may be empty" specifies that empty non-terminals are explicitly recognized. According to the definition of existence in this document, empty non-terminals do exist at the server.

Carefully reading the above paragraph can lead to an interpretation that all possible domains exist - up to the suggested limit of 255 octets for a domain name [RFC 1035]. For example, www.example. may have an A RR, and as far as is practically concerned, is a leaf of the domain tree. But the definition can be taken to mean that sub.www.example. also exists, albeit with no data. By extension, all possible domains exist, from the root on down. As RFC 1034 also defines "an authoritative name error indicating that the name does not exist" in section 4.3.1, this is not the intent of the original document.

RFC1034's wording is to be clarified by adding the following paragraph:

> A node is considered to have an impact on the algorithms of 4.3.2 if it is a leaf node with any resource sets or an interior node, with or without a resource set, that has a subdomain that is a leaf node with a resource set. A QNAME and QCLASS matching an existing node never results in a response return code of authoritative name error.

The terminology in the above paragraph is chosen to remain as close to that in the original document. The term "with" is a alternate form for "owning" in this case, hence "a leaf node owning resources sets, or an interior node, owning or not owning any resource set, that has a leaf node owning a resource set as a subdomain," is the proper interpretation of the middle sentence.

As an aside, an "authoritative name error" has been called NXDOMAIN in

some RFCs, such as RFC 2136 [RFC 2136]. NXDOMAIN is the mnemonic assigned to such an error by at least one implementation of DNS. As this mnemonic is specific to implementations, it is avoided in the remainder of this document.

1.3 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 the document entitled "Key words for use in RFCs to Indicate Requirement Levels." [RFC2119]

Requirements are denoted by paragraphs that begin with with the following convention: 'R'<sect>.<count>.

2 Defining the Wild Card Domain Name

A wild card domain name is defined by having the initial label be:

0000 0001 0010 1010 (binary) = 0x01 0x2a (hexadecimal)

This defines domain names that may play a role in being a wild card, that is, being a source for synthesized answers. Domain names conforming to this definition that appear in queries and RDATA sections do not have any special role. These cases will be described in more detail in following sections.

R2.1 A domain name that is to be interpreted as a wild card MUST begin with a label of '0000 0001 0010 1010' in binary.

The first octet is the normal label type and length for a 1 octet long label, the second octet is the ASCII representation [RFC 20] for the '*' character. In RFC 1034, ASCII encoding is assumed to be the character encoding.

In the master file formats used in RFCs, a "*" is a legal representation for the wild card label. Even if the "*" is escaped, it is still interpreted as the wild card when it is the only character in the label.

R2.2. A server MUST treat a wild card domain name as the basis of synthesized answers regardless of any "escape" sequences in the input format.

RFC 1034 and RFC 1035 ignore the case in which a domain name might be "the*.example.com." The interpretation is that this domain name in a zone would only match queries for "the*.example.com" and not have any other role.

Note: By virtue of this definition, a wild card domain name may have a subdomain. The subdomain (or sub-subdomain) itself may also be a wild card. E.g., *.*.example. is a wild card, so is *.sub.*.example.

More discussion on this is given in Appendix A.

3 Defining Existence

As described in the Introduction, a precise definition of existence is needed.

R3.1 An authoritative server MUST treat a domain name as existing during the execution of the algorithms in RFC 1034 when the domain name conforms to the following definition. A domain name is defined to exist if the domain name owns data and/or has a subdomain that exists.

Note that at a zone boundary, the domain name owns data, including the NS RR set. At the delegating server, the NS RR set is not authoritative, but that is of no consequence here. The domain name owns data, therefore, it exists.

R3.2 An authoritative server MUST treat a domain name that has neither a resource record set nor an existing subdomain as non-existent when executing the algorithm in section 4.3.2. of RFC 1034.

A note on terminology. A domain transcends zones, i.e., all DNS data is in the root domain but segmented into zones of control. In this document, there are references to a "domain name" in the context of existing "in a zone." In this usage, a domain name is the root of a domain, not the entire domain. The domain's root point is said to "exist in a zone" if the zone is authoritative for the name. RR sets existing in a domain need not be owned by the domain's root domain name, but are owned by other domain names in the domain.

4 Impact of a Wild Card Domain In a Query Message

When a wild card domain name appears in a question, e.g., the query name is "*.example.", the response in no way differs from any other query. In other words, the wild card label in a QNAME has no special meaning, and query processing will proceed using '*' as a literal query name.

R4.1 A wild card domain name acting as a QNAME MUST be treated as any other QNAME, there MUST be no special processing accorded it.

If a wild card domain name appears in the RDATA of a CNAME RR or any other RR that has a domain name in it, the same rule applies. In the instance of a CNAME RR, the wild card domain name is used in the same manner of as being the original QNAME. For other RR's, rules vary regarding what is done with the domain name(s) appearing in them, in no case does the wild card hold special meaning.

R4.2 A wild card domain name appearing in any RR's RDATA MUST be treated as any other domain name in that situation, there MUST be no special processing accorded it.

5 Impact of a Wild Card Domain On a Response

The description of how wild cards impact response generation is in RFC 1034, section 4.3.2. That passage contains the algorithm followed by a server in constructing a response. Within that algorithm, step 3, part 'c' defines the behavior of the wild card. The algorithm is directly quoted in lines that begin with a '#' sign. Commentary is interleaved.

[Note that are no requirements specifically listed in this section. The text here is explanatory and interpretative. There is no change to the algorithm specified in RFC 1034.]

The context of part 'c' is that the search is progressing label by label through the QNAME. (Note that the data being searched is the authoritative data in the server, the cache is searched in step 4.) Step 3's part 'a' covers the case that the QNAME has been matched in full, regardless of the presence of a CNAME RR. Step 'b' covers crossing a cut point, resulting in a referral. All that is left is to look for the wild card.

Step 3 of the algorithm also assumes that the search is looking in the zone closest to the answer, i.e., in the same class as QCLASS and as close to the authority as possible on this server. If the zone is not the authority, then a referral is given, possibly one indicating lameness.

```
#          c. If at some label, a match is impossible (i.e., the
#             corresponding label does not exist), look to see if a
#             the "*" label exists.
```

The above paragraph refers to finding the domain name that exists in the zone and that most encloses the QNAME. Such a domain name will mark the boundary of candidate wild card domain names that might be used to synthesize an answer. (Remember that at this point, if the most enclosing name is the same as the QNAME, part 'a' would have recorded an exact match.) The existence of the enclosing name means that no wild card name higher in the tree is a candidate to answer the query.

Once the closest enclosing node is identified, there's the matter of what exists below it. It may have subdomains, but none will be closer to the QNAME. One of the subdomains just might be a wild card. If it exists, this is the only wild card eligible to be used to synthesize an answer for the query. Even if the closest enclosing node conforms to the syntax rule in section 2 for being a wild card domain name, the closest enclosing node is not eligible to be a source of a synthesized answer.

The only wild card domain name that is a candidate to synthesize an answer will be the "*" subdomain of the closest enclosing domain name. Three possibilities can happen. The "*" subdomain does not exist, the "*" subdomain does but does not have an RR set of the same type as the QTYPE, or it exists and has the desired RR set.

For the sake of brevity, the closest enclosing node can be referred to as the "closest encloser." The closest encloser is the most important concept in this clarification. Describing the closest encloser is a bit tricky, but it is an easy concept.

To find the closest encloser, you have to first locate the zone that is the authority for the query name. This eliminates the need to be concerned that the closest encloser is a cut point. In addition, we can assume too that the query name does not exist, hence the closest encloser is not equal to the query name. We can assume away these two cases because they are handled in steps a and b of section 4.3.2.'s algorithm.

What is left is to identify the existing domain name that would have been up the tree (closer to the root) from the query name. Knowing that an exact match is impossible, if there is a "*" label descending from the unique closest encloser, this is the one and only wild card from which an answer can be synthesized for the query.

To illustrate, using the example in section 1.2 of this document, the following chart shows QNAMEs and the closest enclosers. In Appendix A there is another chart showing unusual cases.

QNAME	Closest Encloser	Wild Card Source
host3.example.	example.	*.example.
_telnet._tcp.host1.example.	_tcp.host1.example.	no wild card
_telnet._tcp.host2.example.	host2.example.	no wild card
_telnet._tcp.host3.example.	example.	*.example.
_chat._udp.host3.example.	example.	*.example.

Note that host1.subdel.example. is in a subzone, so the search for it ends in a referral in part 'b', thus does not enter into finding a closest encloser.

The fact that a closest encloser will be the only superdomain that can have a candidate wild card will have an impact when it comes to designing authenticated denial of existence proofs. (This concept is not introduced until DNS Security Extensions are considered in upcoming sections.)

```
#            If the "*" label does not exist, check whether the name
#            we are looking for is the original QNAME in the query
#            or a name we have followed due to a CNAME.  If the name
#            is original, set an authoritative name error in the
#            response and exit.  Otherwise just exit.
```

The above passage says that if there is not even a wild card domain name to match at this point (failing to find an explicit answer elsewhere), we are to return an authoritative name error at this point. If we were following a CNAME, the specification is unclear, but seems to imply that a no error return code is appropriate, with just the CNAME RR (or sequence of CNAME RRs) in the answer section.

```
#            If the "*" label does exist, match RRs at that node
#            against QTYPE.  If any match, copy them into the answer
#            section, but set the owner of the RR to be QNAME, and
#            not the node with the "*" label.  Go to step 6.
```

This final paragraph covers the role of the QTYPE in the process. Note that if no resource record set matches the QTYPE the result is that no data is copied, but the search still ceases ("Go to step 6.").

6 Authenticated Denial and Wild Cards

In unsecured DNS, the only concern when there is no data to return to a query is whether the domain name from which the answer comes exists or not, whether or not a name error is indicated in the return code. In either case the answer section is empty or contained just a sequence of CNAME RR sets.

In securing DNS, authenticated denial of existence is a service that is provided. The chosen solution to provide this service is to generate resource records indicating what is protected in a zone and to digitally sign these.

The resource records that do this, as defined in RFC 2535, are NXT RRs.

There are three points to consider when clarifying the topic of wild card domain names. One is the construction of the records. The second is the inclusion of records in responses. The third is the interpretation of the records in a response by the resolver.

In short, authenticated denial has to be sure to prove that the closest encloser does not equal the query name, whether there is a wild card name directly under the closest encloser.

6.1 Preparing Wild Card Domain Name Owned Non-existence Proofs

During the creation of the authenticated denial records, the wild card domain name plays no special role, in the same manner as the wild card domain name playing no special role in a query.

There are two considerations with regards to preparing non-existence proofs.

R6.1 Any mechanism used to provide authenticated denial MUST reveal the closest enclosing existing domain name for the query. If this is not provided, the resolver will not be able to ascertain the identity of an appropriate wild card domain name.

R6.2 If a zone is signed in such a way that offers authenticated denial of existence, wild card domain name owned RR sets MUST be signed. Otherwise the determination of the "closest encloser" is not possible.

6.2 Role of Wild Cards in Answers

There are three cases to address. The first is synthesizing from wild card domain name with data, the second is negatively synthesizing from an existing wild card, and the third is denying that neither an exact match, referral, nor wild card exist to answer the query.

6.2.1 Synthesizing From a Wild Card

When preparing an answer from a wild card domain name, the answer needs to include proof that the exact match of the QNAME and QCLASS does not exist. This is needed because synthesis of the answer replaces the "*" label with the QNAME without securing the result. The resolver will realize that the answer was derived from a wild card, but cannot detect whether an exact match was maliciously omitted.

R6.3 When synthesizing a positive answer from a wild card domain name, the answer MUST include proof that the exact match for the QNAME and QCLASS does not exist.

Note that a proof that the QTYPE does not exist at the QNAME and QCLASS is not sufficient to justify synthesis from a wild card.

6.2.2. Synthesizing Authoritative No Error, No Data From a Wild Card

When synthesizing a negative answer that is derived from a wild card, meaning that a wild card matched the QNAME (no exact match happened for QNAME) but that there is no match for QTYPE there, at most two negative answers are needed, possibly one. As in 6.2.1, a proof that the exact match failed is needed. A second proof is needed to show that the wild card domain name does not have the QTYPE. Depending on the method of authenticated denial, these this could be possible with one statement.

R6.4 When synthesizing a negative answer from a wild card domain name, the answer MUST include proof that the exact match of the QNAME and QCLASS does not exist and that the QTYPE matches no RR set at the wild card. If this answer can be optimized, an implementation SHOULD reduce the number of records included in the response.

6.2.3. Answering With an Authoritative Name Error

When answering with a result code of a name error, the answer needs to provide proof that neither the exact match for QNAME and QCLASS exists nor that a wild card domain name exists as a subdomain of the closest enclosing domain name.

R6.5 When preparing a reply with an authoritative name error, the answer MUST include proof that the exact match for the QNAME and QCLASS does not exist and that no wild card is available to provide a match.

6.2.4. The Remaining Case (Authoritative No Error, No Data at QNAME)

When answering negatively because there is a match for QNAME and QCLASS but no match for the QTYPE, only a proof for that is needed. Just as the search does not proceed onto a search for the wild card in this case, neither does the construction of the negative answer proof.

R6.6 When preparing a reply in which there is an exact match of the QNAME and QCLASS, but there is no RR set matching the QTYPE, the reply SHOULD NOT contain any proof regarding the wild card domain name.

6.3 Interpreting Negative Answers Involving Wild Cards

There are three requirements for resolvers when it comes to handling negative answers generated as described in section 6.2.

R6.7 A resolver MUST confirm that the negative data relates to the query submitted.

It is incumbent upon the resolver to interpret the answer correctly.

R6.8 A resolver MUST confirm that an answer synthesized from a wild card domain name is done so only in an authoritative absence of a domain name with the query name and query class.

In the case of a wild card synthesized answer, the resolver has to see that the query name and class has no node, proving that a synthesized answer would be appropriate (subject to validation of it).

R6.9 A resolver MUST confirm that an authoritative name error is valid if there is proof that both domain name matching the query name and class and if there is proof that the closest encloser does not have a wild card domain name as an immediate descendent.

Before concluding that an authoritative name error is justified, a resolver has to determine that neither an exact match for the query name and class exists nor an appropriate wild card domain name.

6.4 Authenticated Denial, Wild Card Domain Names, and Opt-In

When considering the Opt-In proposal [WIP], it is wise to not combine a zone that adheres to both opt-in and that has a wild card domain name. The reason is rooted in that the synthesis of an answer is done by substituting the QNAME for the wild card domain name in the answer. Because this is unsecured, and the is ambiguity regarding whether a negative proof can be provided for the exact match (when it is outside the opt-in secured area), a definitive proof of authenticated denial is not possible.

For a more complete discussion of this topic, please refer to the document

describing the Opt-In proposal, referenced above.

7 Analytical Proof That NXT Names the Closest Encloser

How does one know, and (more importantly) *prove* using NXT records, what the closest encloser of a given QNAME is? This section answers that question with a rigorous proof, because security is the topic.

7.1 Background to the Proof

We'd like to have empty non-terminals provably exist in secure zones. In other words, if someone has:

```
a.b.c    3600    IN    A    10.0.0.1
```

in their zone, but does not have any records with owner names "c" or "b.c", we'd like to be able to say (with proof) that "nodes 'c' and 'b.c' exist and yet have no RRs."

We want this because it is the behavior mandated by the nameserver algorithm in section 4.3.2 of RFC 1034, and because it is regarded by most as a better, more "natural" behavior than the alternative of treating such empty non-terminals as being non-existent.

There are two ways to achieve this. One way is to instantiate all the implied empty non-terminals, and then add NXT and SIG(NXT) to them. This works, but is a burden to the server in storage and computation resources. It especially complicates updates, since any deletion of the last record at a node necessitates a computation to determine which empty non-terminals are no longer relevant and thus must also be deleted.

The second way is to infer the existence of the empty non-terminals from the names of the nodes with real data (i.e. the names in the NXT chain).

Using this technique, the "deepest existing ancestor" a.k.a. the "most enclosing name" of any query name Q can be easily found, and proved to exist. This allows great efficiency in the wild card matching algorithm as well, since only one wild card possibility exists and must subsequently be either proven to exist or proven not to exist. This is a big improvement on the "empty non-terminals do not exist" approach, which has many more possible candidate wild card names which must be proven not to exist.

7.2 Definitions and Preliminaries

When we say "subdomain" anywhere below, we mean "is contained within the domain (in the sense that RFC 1034 describes), or is equal to the domain". I.e., we're treating it like "subset" in mathematics.

X is a "superdomain" of Y iff. Y is a subdomain of X.

A name is an "owner name in zone Z" if it is an owner name, is a subdomain of the origin of zone Z, and is not glue (or otherwise beneath a zone cut of zone Z).

A name N is "directly in zone Z" iff. there is some owner name in Z equal to N.

A name N is "inferred to be in zone Z", if it is not directly in zone Z, but is a superdomain of some direct name of Z and is still a subdomain of Z. I.e., it is an "empty non-terminal" required to make the path from the zone origin to some name directly in Z.

A name is "in zone Z" if it is directly in zone Z, or is inferred to be in zone Z.

Let "<" denote the DNSSEC name order relation.

The "greatest common superdomain" of names A and B, denoted GCS(A,B), is the greatest (according to the DNSSEC ordering) name X such that X is a superdomain of both A and B. I.e. it is the "deepest common ancestor" of A and B. GCS(A,B) always exists, because the root name is a superdomain of all names.

Let Q be a name which is a subdomain of the origin of zone Z.

7.3 Bounds of Q in Z

There is always a name directly in Z, call it "GLB(Q,Z)", which is the greatest lower bound of Q. I.e. GLB(Q,Z) <= Q, and for all N in Z where N <= Q, N <= GLB(Q,Z).

There may or may not be a name directly in Z, call it "LUB(Q,Z)", which is the least upper bound of Q. If there is no N directly in Z such that N >= Q, then there is no LUB(Q,Z). If there is some N directly in Z where N >= Q, then there is an LUB(Q,Z) >= Q such that if N >= Q, then LUB(Q,Z) <= N.

So, GLB(Q,Z) <= Q < LUB(Q,Z), if the least upper bound exists.

GLB(Q,Z) will have a NXT record which:

If GLB(Q,Z) = Q, proves that Q is directly in Z

If GLB(Q,Z) != Q, proves that Q is not directly in Z

The "next domain name" field of this NXT record is the LUB, unless it is the zone origin (the DNSSEC "end of chain" marker) and Q != the origin of Z, in which case there is no LUB.

THEOREM 1: Let A, B, and Q be subdomains of Z. Let A <= B and B <= Q. Then

GCS(Q, A) <= GCS(Q, B)

Proof:

Assume GCS(Q, A) > GCS(Q, B). Then A must have more labels in common with Q than B, but since A and B are less than Q, that means that A > B by the DNSSEC ordering, which is a contradiction since A <= B.

THEOREM 2: Let A, B, and Q be subdomains of Z. Let A >= B and B >= Q. Then

GCS(Q, A) <= GCS(Q, B)

Proof:

Assume GCS(Q, A) > GCS(Q, B). Then A must have more labels in common with Q than B, but since A and B are greater than Q, that means that A < B by the DNSSEC ordering, which is a contradiction since A >= B.

7.4 Greatest Ancestor of Q in Z

The "greatest ancestor of Q in Z", denoted GA(Q,Z), is the greatest N in Z, directly or inferred, such that Q is a subdomain of N. GA(Q,Z) is also called the "most enclosing name of Q in Z" or the "deepest ancestor of Q in Z".

GA(Q,Z) always exists. Since Q is a subdomain of the origin of Z, and the origin of Z is "directly in zone Z", so there's always at least one N in Z such that Q is a subdomain of N.

THEOREM 3: Let Q be a subdomain of the origin of zone Z. If LUB(Q,Z) exists, then:

GA(Q,Z) = the greater of GCS(Q, GLB(Q,Z)) and GCS(Q, LUB(Q,Z))

otherwise

GA(Q,Z) = GCS(Q, GLB(Q,Z))

Proof:

We can eliminate the trivial case where Q is directly in Z, since in that case GA(Q,Z) is obviously Q.

For notational convenience, let

L = GCS(Q, GLB(Q,Z))
U = GCS(Q, LUB(Q,Z))

Assume L and U both exist. Assume there is an M in Z that is greater than

both L and U, and is a superdomain of Q.

If M is directly in Z, then M > GLB(Q,Z). This is because if M were <= GLB(Q,Z), then GCS(Q,M) would be <= L by Theorem 1. If M is directly in Z, it cannot be >= Q since it is a superdomain of Q and M != Q. So, we have GLB(Q,Z) < M < Q, which implies that GLB(Q,Z) is not the greatest lower bound, which is a contradiction.

If M is inferred to be in Z, then there is some N directly in Z and M is a superdomain of N. Either N < Q or N > Q (since Q is not directly in Z).

If N < Q, then N > GLB(Q,Z). If N were <= GLB(Q,Z), then the GCS(Q,N) would be <= L by Theorem 1, but GCS(Q,N) = M, and M > L. We thus have a contradiction, since this implies that GLB(Q,Z) is not the greatest lower bound.

If N > Q, then N < LUB(Q,Z). If N were >= LUB(Q,Z), then the GCS(Q,N) would be <= U by Theorem 2, but GCS(Q,N) = M, and M > U. We thus have a contradiction, since this implies that LUB(Q,Z) is not the least upper bound.

Now we deal with the case where U doesn't exist. Again, assume M in Z that is greater than L, and is a superdomain of Q.

The cases where M is directly in Z, or where M is inferred and N < Q are as above. Now we deal with the case where N > Q. First we note that since < is a well-ordering of the names in Z, if there are any upper bounds to Q in Z, then there must be a least upper bound. Now, if N existed, it would be an upper bound of Q in Z, and hence a least upper bound would have to exist, but there is no least upper bound of Q in Z by assumption, so we again have a contradiction.

Q.E.D.

7.5 Conclusion of the Proof

We've shown how to find the "closest encloser" of any given QNAME by looking at the QNAME along with the owner name and "next domain name" field of the NXT record which proves the QNAME doesn't exist. The technique works even when the closest encloser is an inferred name.

Knowing the closest encloser lets us do very simple wild card checking in secure zones, since the only possible matching wild card is

```
*.<closest encloser>
```

We simply lookup that name, and if found, proceed accordingly. If not, we add the NXT record which proves it doesn't exist to the authority section.

8 Security Considerations

This document is refining the specifications to make it more likely that

security can be added to DNS. No functional additions are being made, just refining what is considered proper to allow the DNS, security of the DNS, and extending the DNS to be more predictable.

9 References

Normative References

[RFC 20] ASCII Format for Network Interchange, V.G. Cerf, Oct-16-1969
[RFC 1034] Domain Names - Concepts and Facilities, P.V. Mockapetris, Nov-01-1987
[RFC 1035] Domain Names - Implementation and Specification, P.V Mockapetris, Nov-01-1987
[RFC 2119] Key Words for Use in RFCs to Indicate Requirement Levels, S Bradner, March 1997

Non-normative References

[RFC 2136] Dynamic Updates in the Domain Name System (DNS UPDATE), P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound, April 1997
[RFC 2535] Domain Name System Security Extensions, D. Eastlake, March 1999
[WIP] DNSSEC Opt-In, Internet Draft, R. Arends, M. Kosters, D. Blacka, 2002

10 Others Contributing to This Document

Others who have directly caused text to appear in the document: Paul Vixie and Olaf Kolkman. Many others have indirect influences on the content.

11 Editors

Name: Bob Halley
Affiliation: Nominum, Inc.
Address: 2385 Bay Road, Redwood City, CA 94063 USA
Phone: +1-650-381-6016
EMail: Bob.Halley@nominum.com

Name: Edward Lewis
Affiliation: ARIN
Address: 3635 Concorde Pkwy, Suite 200, Chantilly, VA 20151 USA
Phone: +1-703-227-9854
Email: edlewis@arin.net

Appendix A: Subdomains of Wild Card Domain Names

In reading the definition of section 2 carefully, it is possible to rationalize unusual names as legal. In the example given, *.example. could have subdomains of *.sub.*.example. and even the more direct *.*.example. (The implication here is that these domain names own explicit resource records sets.) Although defining these names is not easy to justify, it is important that implementions account for the possibility. This section will give some further guidence on handling

these names.

The first thing to realize is that by all definitions, subdomains of wild card domain names are legal. In analyzing them, one realizes that they cause no harm by their existence. Because of this, they are allowed to exist, i.e., there are no special case rules made to disallow them. The reason for not preventing these names is that the prevention would just introduce more code paths to put into implementations.

The concept of "closest enclosing" existing names is important to keep in mind. It is also important to realize that a wild card domain name can be a closest encloser of a query name. For example, if *.*.example. is defined in a zone, and the query name is a.*.example., then the closest enclosing domain name is *.example. Keep in mind that the closest encloser is not eligible to be a source of synthesized answers, just the subdomain of it that has the first label "*".

To illustrate this, the following chart shows some matches. Assume that the names *.example., *.*.example., and *.sub.*.example. are defined in the zone.

QNAME	Closest Encloser	Wild Card Source
a.example.	example.	*.example.
b.a.example.	example.	*.example.
a.*.example.	*.example.	*.*.example.
b.a.*.example.	*.example.	*.*.example.
b.a.*.*.example.	*.*.example.	no wild card
a.sub.*.example.	sub.*.example.	*.sub.*.example.
b.a.sub.*.example.	sub.*.example.	*.sub.*.example.
a.*.sub.*.example.	*.sub.*.example.	no wild card
*.a.example.	example.	*.example.
a.sub.b.example.	example.	*.example.

Recall that the closest encloser itself cannot be the wild card. Therefore the match for b.a.*.*.example. has no applicable wild card.

Finally, if a query name is sub.*.example., any answer available will come from an exact name match for sub.*.example. No wild card synthesis is performed in this case.

Full Copyright Statement

Acknowledgement

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

--