Network Working Group
Request for Comments: 4592
Updates: 1034, 2672
Category: Standards Track

E. Lewis
NeuStar

July 2006

The Role of Wildcards in the Domain Name System

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This is an update to the wildcard definition of RFC 1034. The interaction with wildcards and CNAME is changed, an error condition is removed, and the words defining some concepts central to wildcards are changed. The overall goal is not to change wildcards, but to refine the definition of RFC 1034.

Table of Contents

1. Introduction

In RFC 1034 [RFC1034], sections 4.3.2 and 4.3.3 describe the synthesis of answers from special resource records (RRs) called wildcards. The definition in RFC 1034 is incomplete and has proven to be confusing. This document describes the wildcard synthesis by adding to the discussion and making limited modifications. Modifications are made to close inconsistencies that have led to interoperability issues. This description does not expand the service intended by the original definition.

Staying within the spirit and style of the original documents, this document avoids specifying rules for DNS implementations regarding wildcards. The intention is to only describe what is needed for interoperability, not restrict implementation choices. In addition, consideration is given to minimize any backward-compatibility issues with implementations that comply with RFC 1034's definition.

This document is focused on the concept of wildcards as defined in RFC 1034. Nothing is implied regarding alternative means of synthesizing resource record sets (RRSets), nor are alternatives discussed.

1.1. Motivation

Many DNS implementations diverge, in different ways, from the original definition of wildcards. Although there is clearly a need to clarify the original documents in light of this alone, the impetus for this document lay in the engineering of the DNS security extensions [RFC4033]. With an unclear definition of wildcards, the design of authenticated denial became entangled.

This document is intended to limit its changes, documenting only those deemed necessary based on implementation experience, and to remain as close to the original document as possible. To reinforce that this document is meant to clarify and adjust and not redefine wildcards, relevant sections of RFC 1034 are repeated verbatim to facilitate comparison of the old and new text.

1.2. The Original Definition

The definition of the wildcard concept is comprised by the documentation of the algorithm by which a name server prepares a response (in RFC 1034's section 4.3.2) and the way in which a resource record (set) is identified as being a source of synthetic data (section 4.3.3).

This is the definition of the term "wildcard" as it appears in RFC 1034, section 4.3.3.

```
# In the previous algorithm, special treatment was given to RRs with
# owner names starting with the label "*".  Such RRs are called
# wildcards. Wildcard RRs can be thought of as instructions for
# synthesizing RRs.  When the appropriate conditions are met, the
# name server creates RRs with an owner name equal to the query name
# and contents taken from the wildcard RRs.
```

This passage follows the algorithm in which the term wildcard is first used. In this definition, wildcard refers to resource records. In other usage, wildcard has referred to domain names, and it has been used to describe the operational practice of relying on wildcards to generate answers. It is clear from this that there is a need to define clear and unambiguous terminology in the process of discussing wildcards.

The mention of the use of wildcards in the preparation of a response is contained in step 3, part 'c' of RFC 1034's section 4.3.2, entitled "Algorithm". Note that "wildcard" does not appear in the algorithm, instead references are made to the "*" label. The portion of the algorithm relating to wildcards is deconstructed in detail in section 3 of this document; this is the beginning of the relevant portion of the "Algorithm".

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

The scope of this document is the RFC 1034 definition of wildcards and the implications of updates to those documents, such as DNS Security (DNSSEC). Alternate schemes for synthesizing answers are not considered. (Note that there is no reference listed. No document is known to describe any alternate schemes, although there has been some mention of them in mailing lists.)

1.3. Roadmap to This Document

This document accomplishes these three tasks.

- Defines new terms
- Makes minor changes to avoid conflicting concepts
- Describes the actions of certain resource records as wildcards

1.3.1. New Terms

To help in discussing what resource records are wildcards, two terms will be defined: "asterisk label" and "wildcard domain name". These are defined in section 2.1.1.

To assist in clarifying the role of wildcards in the name server algorithm in RFC 1034, section 4.3.2, "source of synthesis" and "closest encloser" are defined. These definitions are in section 3.3.1. "Label match" is defined in section 3.2.

The new terms are used to make discussions of wildcards clearer. Terminology does not directly have an impact on implementations.

1.3.2. Changed Text

The definition of "existence" is changed superficially. This change will not be apparent to implementations; it is needed to make descriptions more precise. The change appears in section 2.2.3.

RFC 1034, section 4.3.3, seems to prohibit having two asterisk labels in a wildcard owner name. With this document, the restriction is removed entirely. This change and its implications are in section 2.1.3.

The actions when a source of synthesis owns a CNAME RR are changed to mirror the actions if an exact match name owns a CNAME RR. This is an addition to the words in RFC 1034, section 4.3.2, step 3, part 'c'. The discussion of this is in section 3.3.3.

Only the latter change represents an impact to implementations. The definition of existence is not a protocol impact. The change to the restriction on names is unlikely to have an impact, as RFC 1034 contained no specification on when and how to enforce the restriction.

1.3.3. Considerations with Special Types

This document describes semantics of wildcard RRSets for "interesting" types as well as empty non-terminal wildcards. Understanding these situations in the context of wildcards has been clouded because these types incur special processing if they are the result of an exact match. This discussion is in section 4.

These discussions do not have an implementation impact; they cover existing knowledge of the types, but to a greater level of detail.

1.4. Standards Terminology

This document does not use terms as defined in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

Quotations of RFC 1034 are denoted by a '#' at the start of the line. References to section "4.3.2" are assumed to refer to RFC 1034's section 4.3.2, simply titled "Algorithm".

2. Wildcard Syntax

The syntax of a wildcard is the same as any other DNS resource record, across all classes and types. The only significant feature is the owner name.

Because wildcards are encoded as resource records with special names, they are included in zone transfers and incremental zone transfers [RFC1995] just as non-wildcard resource records are. This feature has been under appreciated until discussions on alternative approaches to wildcards appeared on mailing lists.

2.1. Identifying a Wildcard

To provide a more accurate description of wildcards, the definition has to start with a discussion of the domain names that appear as owners. Two new terms are needed, "asterisk label" and "wildcard domain name".

2.1.1. Wildcard Domain Name and Asterisk Label

A "wildcard domain name" is defined by having its initial (i.e., leftmost or least significant) label be, in binary format:

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

The first octet is the normal label type and length for a 1-octet-long label, and the second octet is the ASCII representation [RFC20] for the '*' character.

A descriptive name of a label equaling that value is an "asterisk label".

RFC 1034's definition of wildcard would be "a resource record owned by a wildcard domain name".

2.1.2. Asterisks and Other Characters

No label values other than that in section 2.1.1 are asterisk labels, hence names beginning with other labels are never wildcard domain names. Labels such as 'the*' and '**' are not asterisk labels, so these labels do not start wildcard domain names.

2.1.3. Non-terminal Wildcard Domain Names

In section 4.3.3, the following is stated:

```
# ......................... The owner name of the wildcard RRs is
# of the form "*.<anydomain>", where <anydomain> is any domain name.
# <anydomain> should not contain other * labels.....................
```

The restriction is now removed. The original documentation of it is incomplete and the restriction does not serve any purpose given years of operational experience.

There are three possible reasons for putting the restriction in place, but none of the three has held up over time. One is that the restriction meant that there would never be subdomains of wildcard domain names, but the restriction as stated still permits "example.*.example." for instance. Another is that wildcard domain names are not intended to be empty non-terminals, but this situation does not disrupt the algorithm in 4.3.2. Finally, "nested" wildcard domain names are not ambiguous once the concept of the closest encloser had been documented.

A wildcard domain name can have subdomains. There is no need to inspect the subdomains to see if there is another asterisk label in any subdomain.

A wildcard domain name can be an empty non-terminal. (See the upcoming sections on empty non-terminals.) In this case, any lookup encountering it will terminate as would any empty non-terminal match.

2.2. Existence Rules

The notion that a domain name 'exists' is mentioned in the definition of wildcards. In section 4.3.3 of RFC 1034:

```
# Wildcard RRs do not apply:
#
...
#   - When the query name or a name between the wildcard domain and
#     the query name is know[n] to exist. . . .
```

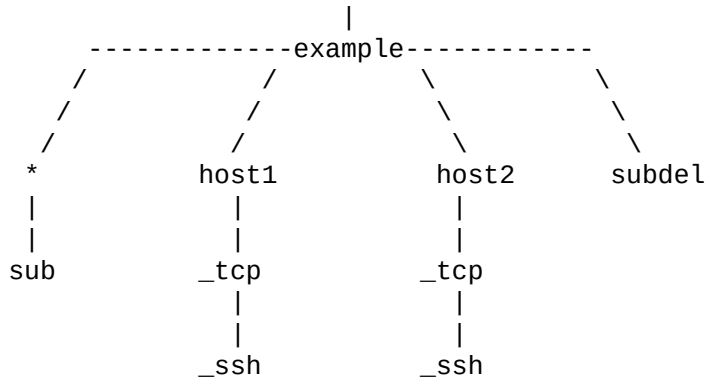
"Existence" is therefore an important concept in the understanding of wildcards. Unfortunately, the definition of what exists, in RFC 1034, is unclear. So, in sections 2.2.2. and 2.2.3, another look is taken at the definition of existence.

2.2.1. An Example

To illustrate what is meant by existence consider this complete zone:

```
$ORIGIN example.
example.                  3600 IN  SOA   <SOA RDATA>
example.                  3600     NS    ns.example.com.
example.                  3600     NS    ns.example.net.
*.example.                3600     TXT   "this is a wildcard"
*.example.                3600     MX    10 host1.example.
sub.*.example.            3600     TXT   "this is not a wildcard"
host1.example.            3600     A     192.0.2.1
_ssh._tcp.host1.example.  3600     SRV   <SRV RDATA>
_ssh._tcp.host2.example.  3600     SRV   <SRV RDATA>
subdel.example.           3600     NS    ns.example.com.
subdel.example.           3600     NS    ns.example.net.
```

A look at the domain names in a tree structure is helpful:



The following responses would be synthesized from one of the wildcards in the zone:

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 ’*.example.’

```
QNAME=foo.bar.example. QTYPE=TXT, QCLASS=IN
     the answer will be "foo.bar.example. IN TXT ..."
     because bar.example. does not exist, but the wildcard
     does.
```

The following responses would not be synthesized from any of the wildcards in the zone:

```
QNAME=host1.example., QTYPE=MX, QCLASS=IN
     because host1.example. exists

QNAME=sub.*.example., QTYPE=MX, QCLASS=IN
     because sub.*.example. exists

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

QNAME=host.subdel.example., QTYPE=A, QCLASS=IN
     because subdel.example. exists (and is a zone cut)

QNAME=ghost.*.example., QTYPE=MX, QCLASS=IN
     because *.example. exists
```

The final example highlights one common misconception about wildcards. A wildcard "blocks itself" in the sense that a wildcard does not match its own subdomains. That is, "*.example." does not match all names in the "example." zone; it fails to match the names below "*.example.". To cover names under "*.example.", another wildcard domain name is needed--"*.*.example."--which covers all but its own subdomains.

2.2.2. Empty Non-terminals

Empty non-terminals [RFC2136, section 7.16] are domain names that own no resource records but have subdomains that do. In section 2.2.1, "_tcp.host1.example." is an example of an empty non-terminal name. Empty non-terminals are introduced by this text 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 and that empty non-terminals "exist".

Pedantically 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 [RFC1035]. 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, so this apparently is not the intent of the original definition, justifying the need for an updated definition in the next section.

2.2.3. Yet Another Definition of Existence

RFC 1034's wording is fixed by the following paragraph:

The domain name space is a tree structure. Nodes in the tree either own at least one RRSet and/or have descendants that collectively own at least one RRSet. A node may exist with no RRSets only if it has descendants that do; this node is an empty non-terminal.

A node with no descendants is a leaf node. Empty leaf nodes do not exist.

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

2.3. When Is a Wildcard Domain Name Not Special?

When a wildcard domain name appears in a message's query section, no special processing occurs. An asterisk label in a query name only matches a single, corresponding asterisk label in the existing zone tree when the 4.3.2 algorithm is being followed.

When a wildcard domain name appears in the resource data of a record, no special processing occurs. An asterisk label in that context literally means just an asterisk.

3. Impact of a Wildcard Domain Name on a Response

RFC 1034's description of how wildcards impact response generation is in its 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 wildcard.

The algorithm in section 4.3.2 is not intended to be pseudo-code; that is, its steps are not intended to be followed in strict order. The "algorithm" is a suggested means of implementing the requirements. As such, in step 3, parts 'a', 'b', and 'c' do not have to be implemented in that order, provided that the result of the implemented code is compliant with the protocol's specification.

3.1. Step 2

Step 2 of section 4.3.2 reads:

```
#    2. Search the available zones for the zone which is the nearest
#       ancestor to QNAME.  If such a zone is found, go to step 3,
#       otherwise step 4.
```

In this step, the most appropriate zone for the response is chosen. The significance of this step is that it means all of step 3 is being performed within one zone. This has significance when considering whether or not an SOA RR can ever be used for synthesis.

3.2. Step 3

Step 3 is dominated by three parts, labeled 'a', 'b', and 'c'. But the beginning of the step is important and needs explanation.

```
#    3. Start matching down, label by label, in the zone.  The
#       matching process can terminate several ways:
```

The word 'matching' refers to label matching. The concept is based in the view of the zone as the tree of existing names. The query name is considered to be an ordered sequence of labels--as if the name were a path from the root to the owner of the desired data (which it is--3rd paragraph of RFC 1034, section 3.1).

The process of label matching a query name ends in exactly one of three choices, the parts 'a', 'b', and 'c'. Either the name is found, the name is below a cut point, or the name is not found.

Once one of the parts is chosen, the other parts are not considered (e.g., do not execute part 'c' and then change the execution path to finish in part 'b'). The process of label matching is also done independent of the query type (QTYPE).

Parts 'a' and 'b' are not an issue for this clarification as they do not relate to record synthesis. Part 'a' is an exact match that results in an answer; part 'b' is a referral.

3.3. Part 'c'

The context of part 'c' is that the process of label matching the labels of the query name has resulted in a situation in which there is no corresponding label in the tree. It is as if the lookup has "fallen off the tree".

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

To help describe the process of looking 'to see if [...] the "*" label exists' a term has been coined to describe the last domain (node) matched. The term is "closest encloser".

3.3.1. Closest Encloser and the Source of Synthesis

The closest encloser is the node in the zone's tree of existing domain names that has the most labels matching the query name (consecutively, counting from the root label downward). Each match is a "label match" and the order of the labels is the same.

The closest encloser is, by definition, an existing name in the zone. The closest encloser might be an empty non-terminal or even be a wildcard domain name itself. In no circumstances is the closest encloser to be used to synthesize records for the current query.

The source of synthesis is defined in the context of a query process as that wildcard domain name immediately descending from the closest encloser, provided that this wildcard domain name exists. "Immediately descending" means that the source of synthesis has a name of the form:

```
<asterisk label>.<closest encloser>.
```

A source of synthesis does not guarantee having a RRSet to use for synthesis. The source of synthesis could be an empty non-terminal.

If the source of synthesis does not exist (not on the domain tree), there will be no wildcard synthesis. There is no search for an alternate.

The important concept is that for any given lookup process, there is at most one place at which wildcard synthetic records can be obtained. If the source of synthesis does not exist, the lookup terminates, and the lookup does not look for other wildcard records.

3.3.2. Closest Encloser and Source of Synthesis Examples

To illustrate, using the example zone in section 2.2.1 of this document, the following chart shows QNAMEs and the closest enclosers.

QNAME	Closest Encloser	Source of Synthesis
host3.example.	example.	*.example.
_telnet._tcp.host1.example.	_tcp.host1.example.	no source
_dns._udp.host2.example.	host2.example.	no source
_telnet._tcp.host3.example.	example.	*.example.
_chat._udp.host3.example.	example.	*.example.
foobar.*.example.	*.example.	no source

3.3.3. Type Matching

RFC 1034 concludes part 'c' with this:

```
#            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.
#
#            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.
```

The final paragraph covers the role of the QTYPE in the lookup process.

Based on implementation feedback and similarities between part 'a' and part 'c', a change to this passage has been made.

The change is to add the following text to part 'c' prior to the instructions to "go to step 6":

> If the data at the source of synthesis is a CNAME, and QTYPE doesn't match CNAME, copy the CNAME RR into the answer section of the response changing the owner name to the QNAME, change QNAME to the canonical name in the CNAME RR, and go back to step 1.

This is essentially the same text in part 'a' covering the processing of CNAME RRSets.

4. Considerations with Special Types

Sections 2 and 3 of this document discuss wildcard synthesis with respect to names in the domain tree and ignore the impact of types. In this section, the implication of wildcards of specific types is discussed. The types covered are those that have proven to be the most difficult to understand. The types are SOA, NS, CNAME, DNAME, SRV, DS, NSEC, RRSIG, and "none", that is, empty non-terminal wildcard domain names.

4.1. SOA RRSet at a Wildcard Domain Name

A wildcard domain name owning an SOA RRSet means that the domain is at the root of the zone (apex). The domain cannot be a source of synthesis because that is, by definition, a descendant node (of the closest encloser) and a zone apex is at the top of the zone.

Although a wildcard domain name owning an SOA RRSet can never be a source of synthesis, there is no reason to forbid the ownership of an SOA RRSet.

For example, given this zone:

```
$ORIGIN *.example.
@               3600 IN  SOA   <SOA RDATA>
                3600     NS    ns1.example.com.
                3600     NS    ns1.example.net.
www             3600     TXT   "the www txt record"
```

A query for www.*.example.'s TXT record would still find the "the www txt record" answer. The asterisk label only becomes significant when section 4.3.2, step 3, part 'c' is in effect.

Of course, there would need to be a delegation in the parent zone, "example." for this to work too. This is covered in the next section.

4.2. NS RRSet at a Wildcard Domain Name

With the definition of DNSSEC [RFC4033, RFC4034, RFC4035] now in place, the semantics of a wildcard domain name owning an NS RRSet has come to be poorly defined. The dilemma relates to a conflict between the rules for synthesis in part 'c' and the fact that the resulting synthesis generates a record for which the zone is not authoritative. In a DNSSEC signed zone, the mechanics of signature management (generation and inclusion in a message) have become unclear.

Salient points of the working group discussion on this topic are summarized in section 4.2.1.

As a result of these discussions, there is no definition given for wildcard domain names owning an NS RRSet. The semantics are left undefined until there is a clear need to have a set defined, and until there is a clear direction to proceed. Operationally, inclusion of wildcard NS RRSets in a zone is discouraged, but not barred.

4.2.1. Discarded Notions

Prior to DNSSEC, a wildcard domain name owning a NS RRSet appeared to be workable, and there are some instances in which it is found in deployments using implementations that support this. Continuing to allow this in the specification is not tenable with DNSSEC. The reason is that the synthesis of the NS RRSet is being done in a zone that has delegated away the responsibility for the name. This "unauthorized" synthesis is not a problem for the base DNS protocol, but DNSSEC in affirming the authorization model for DNS exposes the problem.

Outright banning of wildcards of type NS is also untenable as the DNS protocol does not define how to handle "illegal" data. Implementations may choose not to load a zone, but there is no protocol definition. The lack of the definition is complicated by having to cover dynamic update [RFC2136] and zone transfers, as well as loading at the master server. The case of a client (resolver, caching server) getting a wildcard of type NS in a reply would also have to be considered.

Given the daunting challenge of a complete definition of how to ban such records, dealing with existing implementations that permit the records today is a further complication. There are uses of wildcard domain name owning NS RRSets.

One compromise proposed would have redefined wildcards of type NS to not be used in synthesis, this compromise fell apart because it would have required significant edits to the DNSSEC signing and validation work. (Again, DNSSEC catches unauthorized data.)

With no clear consensus forming on the solution to this dilemma, and the realization that wildcards of type NS are a rarity in operations, the best course of action is to leave this open-ended until "it matters".

4.3. CNAME RRSet at a Wildcard Domain Name

The issue of a CNAME RRSet owned by a wildcard domain name has prompted a suggested change to the last paragraph of step 3c of the algorithm in 4.3.2. The changed text appears in section 3.3.3 of this document.

4.4. DNAME RRSet at a Wildcard Domain Name

Ownership of a DNAME [RFC2672] RRSet by a wildcard domain name represents a threat to the coherency of the DNS and is to be avoided or outright rejected. Such a DNAME RRSet represents non-deterministic synthesis of rules fed to different caches. As caches are fed the different rules (in an unpredictable manner) the caches will cease to be coherent. ("As caches are fed" refers to the storage in a cache of records obtained in responses by recursive or iterative servers.)

For example, assume one cache, responding to a recursive request, obtains the following record:

"a.b.example. DNAME foo.bar.example.net."

and another cache obtains:

"b.example. DNAME foo.bar.example.net."

both generated from the record:

"*.example. DNAME foo.bar.example.net."

by an authoritative server.

The DNAME specification is not clear on whether DNAME records in a cache are used to rewrite queries. In some interpretations, the rewrite occurs; in others, it does not. Allowing for the occurrence of rewriting, queries for "sub.a.b.example. A" may be rewritten as "sub.foo.bar.tld. A" by the former caching server and may be rewritten as "sub.a.foo.bar.tld. A" by the latter. Coherency is lost, and an operational nightmare ensues.

Another justification for a recommendation to avoid the use of wildcard DNAME records is the observation that such a record could synthesize a DNAME owned by "sub.foo.bar.example." and "foo.bar.example.". There is a restriction in the DNAME definition that no domain exist below a DNAME-owning domain; hence, the wildcard DNAME is to be avoided.

4.5. SRV RRSet at a Wildcard Domain Name

The definition of the SRV RRset is RFC 2782 [RFC2782]. In the definition of the record, there is some confusion over the term "Name". The definition reads as follows:

```
# The format of the SRV RR
...
#    _Service._Proto.Name TTL Class SRV Priority Weight Port Target
...
#  Name
#   The domain this RR refers to.  The SRV RR is unique in that the
#   name one searches for is not this name; the example near the end
#   shows this clearly.
```

Do not confuse the definition "Name" with the owner name. That is, once removing the _Service and _Proto labels from the owner name of the SRV RRSet, what remains could be a wildcard domain name but this is immaterial to the SRV RRSet.

For example, if an SRV record is the following:

```
   _foo._udp.*.example. 10800 IN SRV 0 1 9 old-slow-box.example.
```

.example is a wildcard domain name and although it is the Name of the SRV RR, it is not the owner (domain name). The owner domain name is "_foo._udp..example.", which is not a wildcard domain name.

A query for the SRV RRSet of "_foo._udp.bar.example." (class IN), will result in a match of the name "*.example." (assuming there is no bar.example.) and not a match of the SRV record shown. If there is no SRV RRSet at "*.example.", the answer section will reflect that (be empty or a CNAME RRset).

The confusion is likely based on the mixture of the specification of the SRV RR and the description of a "use case".

4.6. DS RRSet at a Wildcard Domain Name

A DS RRSet owned by a wildcard domain name is meaningless and harmless. This statement is made in the context that an NS RRSet at a wildcard domain name is undefined. At a non-delegation point, a DS RRSet has no value (no corresponding DNSKEY RRSet will be used in DNSSEC validation). If there is a synthesized DS RRSet, it alone will not be very useful as it exists in the context of a delegation point.

4.7. NSEC RRSet at a Wildcard Domain Name

Wildcard domain names in DNSSEC signed zones will have an NSEC RRSet. Synthesis of these records will only occur when the query exactly matches the record. Synthesized NSEC RRs will not be harmful as they will never be used in negative caching or to generate a negative response [RFC2308].

4.8. RRSIG at a Wildcard Domain Name

RRSIG records will be present at a wildcard domain name in a signed zone and will be synthesized along with data sought in a query. The fact that the owner name is synthesized is not a problem as the label count in the RRSIG will instruct the verifying code to ignore it.

4.9. Empty Non-terminal Wildcard Domain Name

If a source of synthesis is an empty non-terminal, then the response will be one of no error in the return code and no RRSet in the answer section.

5. 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.

6. References

6.1. Normative References

[RFC20] Cerf, V., "ASCII format for network interchange", RFC 20, October 1969.

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

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

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, August 1996.

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

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC2672] Crawford, M., "Non-Terminal DNS Name Redirection", RFC 2672, August 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

6.2. Informative References

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

7. Others Contributing to the Document

This document represents the work of a large working group. The editor merely recorded its collective wisdom.

Comments on this document can be sent to the editor or the mailing list for the DNSEXT WG, namedroppers@ops.ietf.org.

Editor's Address

Edward Lewis
NeuStar
46000 Center Oak Plaza
Sterling, VA
20166, US

Phone: +1-571-434-5468
EMail: ed.lewis@neustar.biz

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).