DNSEXT (Independent submission)
Internet-Draft
Expires: July 4, 2003

O. Kolkman
RIPE NCC
J. Ihren
Autonomica
R. Arends
Telematica Instituut
January 3, 2003

DNSSEC Wildcard Optimization
draft-olaf-dnsext-dnssec-wildcard-optimization-02.txt

Status of this Memo

Copyright Notice

Abstract

Secure denial of the existence of wildcards may lead to a large number of NXT Resource Records and associated SIG Resource Records in DNS responses, even in the common case when wildcards are not present in the zone. This optimization uses one bit from the NXT type array to signal that there is no closer wildcard in the zone for a given query name. This reduces the packet size and the need for executing slow, and complicated, code paths in the case when queries are made to zones which have the bit set at zone signing time. In cases where

there are no wildcard RRs in the zone (e.g. the root zone) only one NXT RR and corresponding SIG is needed for denial of existence of both a full match and any possible wildcard matches.

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

Table of Contents

1. Introduction

Wildcards make authenticated denial of existence complex. Many zones do not contain wildcards but still incur a penalty. If the NXT RR contains an indication that a wildcard match can not exist then less DNSSEC related RRs and less computation are needed to authoritatively deny the existence of a name in the zone.

1.1 RFC2535 Wildcard Processing

RFC2535 [3] specifies that the non-existence of a match against a wildcard is proven by a set of relevant NXT records. In practice this will result to at least 2 NXT RRs and corresponding SIGs being returned. Even in zones that do not use wildcards this will lead to complex answers for which the resolvers will need to follow NXT chains and which are hard to troubleshoot by operators.

1.2 Optimizations

With the solution proposed herein the following optimizations are realized:

- Both servers and resolvers can bypass slow and complicated code paths in the common case where no wildcards are present in a secured zone.

- Packet size of answers reduce in most common cases; for the root zone the authority section only contains one NXT RR with associated SIGs instead of two NXT RRs with associated SIGs.

- The common case answers are easier to interpret by human operators troubleshooting responses;

1.3 Complexities

With the solution proposed herein the following complexities are added:

- Increased code complexity; the proposed optimization leads to a small increase in the amount of code i.e. a few conditional statements branching off the algorithm proofing the non-existence of wildcard.

Note that authoritative server code can be designed without support for secured wildcard RRs, these servers will not have the added complexity but will also not be able to serve zones with wildcards.

2. Definition of the NOWILD bit

The NXT RR, used to the prove the non-existence of data, uses a type bit-map to track which types are available for a given name. We propose to use one bit (see section Section 4) in the type bitmap to signal that no wildcard match is possible for the part of the the zone's namespace covered by the NXT RR. We refer to this bit as the "NOWILD-bit".

If the NOWILD-bit in a NXT RR is set to 1 then there is no wildcard expansion possible for any of the possible names between and including the ownername and the NXT-dname of that NXT RR.

By setting the bit the zone-owner makes a statement about the non-existence of possible wildcards. It is important that the zone owner applies the proper algorithm to set the bit. Two possible algorithms are presented below.

2.1 Algorithms to set the bit

Any algorithm that sets the NOWILD bit in the NXT RRs in a zone MUST NOT set the bit if a wildcard match is possible for any of the possible names between and including the ownername and the NXT-dname of a NXT RR. Any algorithm MAY set the bit if the negation of the above is true i.e. where there is no wildcard expansion possible for any of the names spanned by the NXT RR.

When an algorithm as above is applied a NXT RR that proves the non-existence of a full match of the QNAME will also prove, when it's NOWILD-bit is set to 1, that there is no match of the QNAME to any wildcard that may exist in the zone

2.1.1 Trivial Algorithm

The most trivial algorithm is to set the bit in all NXT RRs in a zone if there are no wildcards in that zone and to clear it if there is any wildcard in the zone.

2.1.2 'Advanced' Algorithm

Now for a somewhat complicated algorithm.

Represent the ownername and the NXT-dname in a set of labels like: 'label(j).label(j-1).label(j-2) ... label(0).'. The NOWILD bit can than be set if for both the ownername and the NXT-dname there is no wildcard name '*.label(i).label(i-1) ... label(0).' in the zone for all i < j. In other words the NOWILD is set to 0 if -- while ignoring the possible existence of a domain name between the

ownername and the wildcard domain -- there exists a wildcard that would match QNAME=ownername or QNAME=NXT-dname.

For ownernames for which the above does not apply the NOWILD bit can be set to 1.

3. DNSSEC Protocol Changes

This is an update to the RFC2535 protocol. Resolvers MUST implement these changes. Servers MAY implement these changes.

3.1 Server Side

3.1.1 Zone Signing

Servers that implement the optimization MAY perform the following actions at zone signing time.

At zone signing time an algorithm that sets the bit according to the above definition.

If, because of implementation or policy issues, the algorithm is not applied then the bit MUST be set to 0 for all NXT RRs in the zone. Servers that do do not implement the optimization have already set their NOWILD bit to 0 by virtue of the requirements of RFC2535 section 5.2.

3.1.2 Server Responses

When queried for a name for which there is no match, i.e. no full and no wildcard match, in the zone:

- Servers MUST return the NXT RR that proves the non-existence of the query name in the NXDOMAIN response. If there is no match for a wildcard and the NOWILD-bit is set to 1 at signing time and the one NXT RR is sufficient. If the NOWILD-bit for the NXT RR that proves non-existence of the query name is set to 0 then NXT RRs that prove the non-existence of possible wildcard matches MUST be returned as well.

When queried for a name for which there is a match in the zone:

- If the match is an exact match than no NXT RRs are returned in the additional section.

- Servers for zones that contain one or more wildcards MUST return the NXT RRs that prove the non-existence of the exact match. They must also provide proof that there is no closer match for the

QNAME than the match returned in the answer section.

This proof algorithm for non-existence of wildcards, an exact match or closer matches conforms to RFC2535.

3.1.3 Dynamic DNS considerations

When dynamically adding or removing a name that does not contains wildcards, the 'next name' for the name immediately above the inserted, or deleted name needs to be updated. The NOWILD bit of the inserted name is to be set according to the algorithm as described in Section 2.1. Except for setting the NOWILD bit this is similar to the RFC2535 procedure.

If a name containing a wildcard is deleted from a zone one has to verify if, for all names in the zone with the bit set to 0, the NOWILD bit can be toggled. If a name containing a wildcard is added one has to verify if, for all the names in the zone, the bit needs to be set to 0.

The NOWILD bit is not to be modified during an update of a name that already exists in the zone.

Dynamic updates of names that contain wildcards may lead to performance penalties for large dynamic zones and one may therefore choose not to perform the NOWILD optimization for dynamic zones.

3.2 Resolver Side

When receiving an answer to a query a resolver MUST assess if the answer is a result of a wildcard match. If the result is an exact match then there will be no NXT RRs in the authority section.

If the answer is a wildcard match then the resolver will need to verify that the exact name does not exist. The NXT RRs in the additional section, which per definition have their NOWILD-bit set to 0, will need to prove that there is no closer match. (conforming to RFC2535).

If the response is NXDOMAIN (i.e. no match at all) then the resolver MUST verify if the NXT RR proves the non-existence of the exact match in the zone. No further NXT RRs are needed if the NXT RR has it's NOWILD-bit set to 1. A DNS packet containing an NXDOMAIN response accompanied by a NXT RR that has it's NOWILD-bit set to 0 will need to contain proof that there are no wildcard matches against the QNAME (conforming to RFC2535).

3.2.1 RFC2535 compatibility

A verifier, that does not have this technology implemented, will not obtain sufficient proof to assess the non existence of wildcard from servers that have implemented the optimizations. Although it is possible for a verifier to obtain the proof by a number of additional queries it can not be expected that a verifier will actually do that.

This optimization is therefore incompatible with RFC2535 style denial of existence.

4. IANA Considerations

Although there is no RR record associated the NOWILD-bit. The value of the bit must be registered as a DNS RR-type. To not cause the NXT type bitmap to grow beyond 4 octets unnecessarily we propose to reuse type code 31 (the EID type code is undocumented).

5. Security Considerations

The draft provides an optimization for wildcard handling. Resolvers MUST verify the denial of existence of matches or the denial of existence of closer matches when an answer is returned and the NOWILD-bit is set to 0.

6. Internationalization Considerations

There are no internationalization considerations.

7. Acknowledgements

Olafur Gudmundsson, Daniel Karrenberg, Matt Larson, Ed Lewis, Ted Lindgreen, Daniel Massey, Scott Rose and Sam Weiler for providing critique and input on earlier versions of this document.

8. Document Changes

8.1 draft 00->01

Reordered and reworded the 'protocol changes' section. We tried to make the fact that resolvers must and servers may implement this optimization more explicit.

Change from using the SIG bit to another bit in the NXT type-bitmap, changed the name of the bit and added IANA considerations. Note that the meaning of the bit being set and cleared are changed because of the default setting. Because of the fact that we want to maintain backward compatibility with servers that do not

implement this bit and the bit in the typemap is currently set to 0 the default behaviour should be follow old-style NXT proof.

Corrected mistakes in the examples.

Various style and spelling corrections.

8.2 draft 01->02

Restructured the document somewhat by explicitly defining the meaning from the bit and describing (some of the algorithms) to set the bit in a separate section.

The algorithm described in 00 and 01 was broken for the case that there are multiple labels sorting before the '*'.

Sanitized the examples. The existence of a 'non-terminating' label in the NXT response often (maybe always) shortcuts possible matches between the QNAME and other relevant possible wildcard entries (See Appendix A.1.2).

Normative References

[1] Gudmundsson, O., "Delegation Signer Resource Record", draft-ietf-dnsext-delegation-signer-12 (work in progress), December 2002.

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

[3] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

Authors' Addresses

Olaf M. Kolkman
RIPE NCC
Singel 256
1016 AB Amsterdam
NL

Phone: +31 20 535 4444
EMail: olaf@ripe.net
URI: http://www.ripe.net/

Johan Ihren
Autonomica
Bellmansgatan 30
SE-118 47 Stockholm
SE

EMail: johani@autonomica.se

Roy Arends
Telematica Instituut
Drienerlolaan 5
7522 NB Enschede
NL

EMail: roy.arends@telin.nl

Appendix A. Examples

A.1 Zone without wildcards

In the following example zone file there are no wildcards. All NOWILD bits are set to 1. The actual SIG RRs and the KEY RRs are left out from the zone data and type bitmaps for clarity only.

```
$ORIGIN example.
@ IN   SOA
@      NXT a     SOA NXT NOWILD     ; NOWILD-bit set to 1
a      A         10.0.0.1
a      NXT a.b   A NXT NOWILD       ; NOWILD-bit set to 1
a.b    A         10.0.0.2
a.b    NXT a.c   A NXT NOWILD       ; NOWILD-bit set to 1
a.c    A         10.0.0.4
a.c    NXT a.b.c A NXT NOWILD       ; NOWILD-bit set to 1
a.b.c  A         10.0.0.5
a.b.c  NXT f     A NXT NOWILD       ; NOWILD-bit set to 1
f      A         10.0.0.6
f      NXT @     A NXT NOWILD       ; NOWILD-bit set to 1
```

A.1.1 Optimized proof

A query for any existing name will return a signed answer without NXT RRs in the authority section. A query for any non existing name will only return 1 NXT RR proving the non-existence of the QNAME in the zone and, by virtue of the NOWILD-bit being 1, this is sufficient proof there is no wildcard.

```
QNAME= d.b.c.example. QTYPE=A

RCODE=NXDOMAIN
;; Authority
example.          SOA
                  SIG SOA
a.b.c.example.    NXT f.example.    A NXT NOWILD
                  SIG NXT
;; Additional
(... skipped ... )
```

A.1.2 RFC2535 proof

For comparison we supply the same answer without the optimization applied i.e. NOWILD set to 0 for all NXT RRs in the zone. The answer needs to contain prove that *.b.c.example, *.c.example and *.example do not exist, unless a name that exists in the zone terminates the possible match of those wildcards against the QNAME.

```
QNAME= d.b.c.example. QTYPE=A

RCODE=NXDOMAIN
;; Authority
example.          SOA
                  SIG SOA
a.b.c.example.    NXT f.example. A NXT
                  SIG NXT
                  ; proofs non-existence of exact match.

a.c.example.      NXT a.b.c.example. A NXT
                  SIG NXT
                  ; proofs non-existence of  *.b.c.example.


;; Additional
(... skipped ... )
```

Note that the existence of 'a.b.c.example NXT' RR terminates a wildcard match of QNAME against *.c.example. and *.example. So the answer packet does not need to contain further proof for the non-existence of those wildcards. However, a resolver will have to execute logic to verify that the existence of 'a.b.c.example.' terminates the possible match of the QNAME against the possible wildcards and that the answer is therefore complete.

A.2 Zone with wildcards

In the following example zone file there is a wildcard. Some NOWILD bits are set to 1, others for which there is no wildcard in the zone if the leftmost labels are chopped off, have there NOWILD-bit set to 0. The actual SIG RRs and the KEY RRs at the apex are left out for clarity. The queries for which a wildcard match is returned will have the NOWILD-bit set to 0, there proof for the non-existing closer match is to be supplied and checked by the resolver.

```
$ORIGIN example.
@ IN   SOA
@      NXT a      SOA NXT SIG NOWILD ; NOWILD-bit set to 1
a      A          10.0.0.1
a      NXT a.b    A NXT SIG NOWILD   ; NOWILD-bit set to 1
a.b    A          10.0.0.2
a.b    NXT *.c    A NXT SIG NOWILD   ; NOWILD-bit set to 1
*.c    A          10.0.0.3
*.c    NXT a.c    A NXT SIG          ; NOWILD-bit set to 0
a.c    A          10.0.0.4
a.c    NXT a.b.c  A NXT SIG          ; NOWILD-bit set to 0
a.b.c  A          10.0.0.5
a.b.c  NXT f      A NXT SIG          ; NOWILD-bit set to 0
f      A          10.0.0.6
f      NXT @      A NXT SIG NOWILD   ; NOWILD-bit set to 1
```

In the above example 'a.b.c NXT f' has NOWILD=0 because a.b.c is a possible expansion of '*.c'. 'a.b NXT *.c' has NOWILD=0 because there are names in the namespace that is spanned by the NXT RR that match the *.c wildcard. For instance '!.c', a valid domain name which sorts canonically before '*.c' and after 'a.b'.

A.2.1 Optimized proof

```
QNAME= c.a.a.example. QTYPE=A

RCODE=NXDOMAIN
;; Authority
example.       SOA
               SIG SOA
a.example.     NXT a.b.example.    A NXT SIG NOWILD
                                  ; NOWILD-bit set to 1 proves no full
                                  ; match and no wildcards that match
                                  ; QNAME
               SIG NXT


;; Additional
(... skipped ... )
```

A.2.2 NXDOMAIN with additional proof for no wildcard

The following example contains a NXDOMAIN answer and the proof that there is no wildcard match. The NXT RR proving the non-existence of a matching wildcard needs to be supplied because the NOWILD bit is not set on the entry that proves an exact match

```
QNAME= e.example. QTYPE=A

RCODE=NXDOMAIN
;; Authority
example.example  SOA
                 SIG SOA
a.b.c.example.   NXT f.example.    A NXT SIG      ; NOWILD-bit set to 0,
                                                  ; proves no full match
                 SIG NXT
example.         NXT a.example A NXT SIG NOWILD   ; NOWILD-bit set to 1,
                                                  ; proves no *.example.


;; Additional
(... skipped ... )
```

A.2.3 Another Optimized Proof

The following example contains a NXDOMAIN answer and the proof that there is no wildcard match. In this particular case the proof is optimized because of the NOWILD-bit on the f NXT RR being set to

zero.

```
 QNAME= g.example. QTYPE=A

 RCODE=NXDOMAIN
 ;; Authority
 example.example  SOA
                  SIG SOA
 f.example.       NXT example.   A NXT NOWILD    ; NOWILD-bit set to 1
                                                 ; proves no full match

 ;; Additional
 (... skipped ... )
```

A.2.4 Denial of Existence of Closer Match

The following example contains an answer with wildcard expansion and the proof that there is no closer match. This is similar to a RFC2535 proof of non-existence.

```
 QNAME= d.b.c.example. QTYPE=A

 RCODE=ANSWER
 ;; Answer
 d.b.c.example.   A    10.0.0.3                  ; expansion of *.c
                  SIG A  (labelcount=2)          ; labelcount proofs
                                                 ; wildcard expansion
 ;; Authority
 example.example. SOA
                  SIG SOA
 a.b.c.example.   NXT f.example.    A NXT SIG    ; NOWILD-bit set to 0,
                                                 ; proves no exact match,
                  SIG  NXT
 a.c.example.     NXT a.b.c.example. A NXT SIG   ; NOWILD-bit set to 0
                                                 ; proves non-existence of
                                                 ; *.b.c.example.
                                                 ; No further proofs needed

 ;; Additional
 (... skipped ... )
```

Full Copyright Statement

Acknowledgement

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