Workgroup: Network Working Group
Published: 8 September 2022
Intended Status: Informational
Expires: 12 March 2023
Authors: J. Levine
Taughannock Networks

Publishing Organization Boundaries in the DNS

Abstract

The organization that manages a subtree in the DNS is often different from the one that manages the tree above it. We describe an architecture to publish in the DNS the boundaries between organizations that can be adapted to various policy models and can be queried with a small number of DNS lookups.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Often, the organization that manages a subtree in the DNS is different from the one that manages the tree above it. Many applications use information about such boundaries to implement security policies. For example, web browsers use them to limit the names where HTTP cookies can be set, and Secure Socket Layer (SSL) certificate services use them to determine the party responsible for the domain in a signing request. Mail security applications such as Domain-based Messaging Authentication, Reporting and Conformance (DMARC) use them to locate an organization's policy records in the DNS. This specification is intended to provide boundaries usable for DMARC, and possibly for other applications.

[[Please direct discussion of this draft to the dbound mailing list at dbound@ietf.org.]]

2. Design Issues

Organization boundaries can be assigned on what one could call an opt-in or opt-out basis. "Opt-in" means that two names are only managed by the same organization if both actively assert that they are related. "Opt-out" means that if there is any boundary information at all for a DNS subtree, each name is assumed to be under the same management as its parent unless there is a boundary assertion to the contrary. This design describes an opt-out model.

Within the opt-out model, this design can adapt to a variety of scenarios:

* Policies can be published by the domains themselves, or by a third party. In the former case, each domain might assert its own boundary policies. In the latter case, the third party makes the assertions, which may or may not agree with what the domains themselves would want.

* Multiple levels of delegation may be implemented, which is different from irregular boundaries. For example, "us", "ny.us", and "k12.ny.us" are irregular boundaries, because they're all handled by the US top-level domain registry operator. CentralNIC's "uk.com" would be a second level of delegation below Verisign's com.

* Different sets of boundary rules can be published for different applications. For example, the boundaries for SSL certificates might be different from the boundaries for e-mail policies, or for HTTP cookie setting policies.

In the lookup process below, the boundary point data is stored in the DNS tree in a TXT record. The boundary is considered to be directly below the name that the process returns, similarly to the names in the PSL [PSL]. If the process returned "abc.example", then "foo.abc.example" and "bar.abc.example" are separated by the boundary, but "foo.abc.example" and "bar.foo.abc.example" are not.

Each domain can publish its own policies within its own domain name space, or a separate authority can publish a global set of policies in a separate name space.

2.1. Non-goals

This specification is intended only to describe vertical relationships between domain names and their ancestors or descendants. For example, if there is a boundary between "com" and "example.com", but no boundary between "example.com" and "www.example.com", that indicates that "com" is one organization, while "example.com" and "www.example.com" are a different organization.

While it may well be useful to indicate that "example.com" and "example.net" are the same organization, this specification provides no way to describe horizontal or cross-tree relationships.

This specification deliberately says nothing about zone cuts or zone boundaries. While some zone cuts may match organization boundaries, they often do not. It is quite common to have multiple zones within an organization for administrative convenience, or for a hosting provider to put the names of many customers' hosts in a single zone.

3. TXT record format

```
 *._bound.k12.ny.us IN TXT "bound=1 . . k12.ny.us"
```

The bound TXT record contains a text string with four fields separated by a single space: a tag, two keyword fields and a domain name.

The "bound=1" tag is to prevent confusion when a domain publishes a wildcard such as *.example.com that could match a _bound name. Records that do not start with the correct tag or that do not have four space separated fields are ignored.

Each keyword field is a series of comma separated keywords. If the field would otherwise be empty, it is a single dot. The keywords are listed in IANA registries (Section 11).

The first keyword field expresses policy options. It can include NOLOWER which means that no lower level boundaries can exist below this one, and NOBOUND which means that this name is not a boundary for this application.

The second keyword field identifies the application(s) to which this boundary applies. The keywords DMARC, COOKIE, and CERT mean that the applications are DMARC, HTTP cookies, and SSL certificate signing respectively; a dot means it is a default for any applications not otherwise specified.

The domain name is an absolute domain name, without the final dot.

4. Lookup Process

In general, the lookup process takes as input a domain name and application. It returns the name of the boundary node in the DNS. This may be the domain itself or a parent. If there is no policy for the domain the lookup fails; there are no default boundaries. (Applications may apply defaults of their own, but that is beyond the scope of this specification.)

Names of boundary information records use the tag "_bound" which is intended to be unique.

For the first lookup, the client extracts the top level component (i.e., the rightmost label, as "label" is defined in Section 3 of

[RFC1034]) of the domain name from the subcomponents, if any, and inserts the prefix in front of that component, after other components if any. For example, if the domain to be checked is "example.com" or "www.example.com", the client issues a DNS query for "example._bound.com" or "www.example._bound.com". If the domain is a dotless one such as "example", the client looks up "_bound.example".

Then the client does a DNS lookup of TXT records at that name, which will return zero or more TXT records. A failure such as NXDOMAIN is considered to return zero records. A lookup can return multiple records if different applications have different boundaries or policy options. The lookup process discards any records that do not start with "bound=1" or contain less than four strings.

If a relevant policy record is returned, and the record does not contain the NOBOUND keyword, the domain name in the record is the policy boundary. A policy record is relevant if it lists the desired application, or it is a default policy and there is no record with the application's keyword. For example, a check for a boundary above "example.com" would be issued at "example._bound.com", and the expected TXT record could be "bound=1 . . com".

If there are no boundaries at all in a TLD, the policy record contains "bound=1 . . ." indicating the root. For example, if all subdomains of the "example" top-level domain (TLD) are under the same management as the TLD itself, checks for "_bound.example" or "www._bound.example" would return "bound=1 . . .".

If the relevant record has the NOLOWER keyword set, the process stops. Otherwise, the client inserts the prefix tag into the name just below (i.e., to the left of) the name at the largest matching boundary indicated in the lookup result, and repeats the lookup. For example:

* When evaluating www.foo.example.com, the first query would be to www.foo.example._bound.com. If the reply to this is "bound=1 . . com", then the second query would be to www.foo._bound.example.com.

* When evaluating www.example.ny.us, the first query would be to www.example.ny._bound.us. If the reply to this is "bound=1" . . us", the next lookup would be to www.example._bound.ny.us. If it returned "bound=1 . . ny.us", the third lookup would be to www._bound.example.ny.us.

This process repeats until a DNS lookup returns a relevant record with the NOLOWER keyword, or a lookup returns no relevant records, at which point the boundary is the domain name in the last retrieved relevant record.

If an otherwise relevant record has the NOBOUND keyword, the process continues if the NOLOWER keyword is not present, but there is no boundary at the name with the NOBOUND keyword. The NOBOUND keyword enables a name in the hierarchy to be a boundary for some applications but not for others. A record might have NOLOWER,NOBOUND if it is at a name that is a boundary for some applications but not others, and has no boundaries below it.

5. DNS Records

The publishing entity uses wildcards and prefixed names that parallel the regular names under a TLD to cover the domain's name space.

If there is a boundary at a given name, an entry in the TLD record covers the names below it. For example, if there is a boundary at ".TEST", a suitable record would be:

```
*._bound.test IN TXT "bound=1 . . test"
```

If the boundary is above the TEST domain, i.e., TEST is under the same management as FOO.TEST, the record would indicate no boundaries, and an additional non-wildcard record is needed to cover TEST itself:

```
*._bound.test IN TXT "bound=1 . . ."
_bound.test   IN TXT "bound=1 . . ."
```

In domains with irregular policy boundaries, multiple records in the record describe the boundary points. For example, in the US (United States) TLD, there are legacy domains under XX.US where XX is a two-letter state abbreviation, and there are some further points such as "k12" for schools within a state, with a boundary such as such as "k12.ny.us". A suitable set of of records can cover this structure. The closest encloser rule in RFC 4592 [RFC4592] makes the wildcards match the appropriate names.

```
*._bound.us            IN TXT "bound=1 . . us"
*._bound.ny.us         IN TXT "bound=1 . . ny.us"
*._bound.k12.ny.us     IN TXT "bound=1 . . k12.ny.us"
```

In the usual case that only the boundary closest to the looked up domain name is of interest, the publishing entity can publish "shadow" wildcard records for lower level boundaries that are used rather than the higher level records for names below those boundaries:

```
*.ny._bound.us         IN TXT "bound=1 . . ny.us"
ny._bound.us           IN TXT "bound=1 . . ny.us"
*.k12.ny._bound.us     IN TXT "bound=1 . . k12.ny.us"
k12.ny._bound.us       IN TXT "bound=1 . . k12.ny.us"
```

Each shadow record also needs to be matched by a similar record without the wildcard, since the non-wildcard name would otherwise be an empty non-terminal which wildcard lookups don't match, so a lookup for that name would return nothing at all.

For any set of policy boundaries in a tree of DNS names, a suitable set of policy records can describe the boundaries, so a client can find the boundary for any name in the tree with a single policy lookup per level of delegation.

Since the delegation structure is unlikely to change frequently, long time-to-live (TTL) values in the TXT records are appropriate.

If different applications have different boundaries or policy options, the policy records for each application are put at the appropriate names for the boundaries. Due to the way DNS wildcards work, each name with any policy records **MUST** have records for all policies, with the NOBOUND bit for policies for which the name is not in fact a boundary. If this is the lowest boundary in the DNS subtree, all of the records have NOLOWER. In the example below, there is a boundary at abc.example.com for DMARC but not for any other application.

```
*._bound.abc.example.com IN TXT "bound=1 . DMARC abc.example.com"
*._bound.abc.example.com IN TXT "bound=1 NOBOUND . abc.example.com"
```

6. Application scenarios

Here are some ways that DMARC and potentially other applications can use BOUND data.

6.1. DMARC

If a DMARC lookup for the domain in a message's From: header fails, the client would do a boundary check for the domain name using the "DMARC" application. The organizational domain is the immediate subdomain of the boundary domain. (Note that the boundary will always be the one looked up or an ancestor.)

6.2. Cookies

If an http request attempts to set a cookie for a domain other than the request's own domain, the client would do boundary check for a "COOKIE" application for both the request's domain and the cookie domain. If they are not separated by a boundary, the request is allowed.

6.3. SSL Certificates

The client would do a boundary check for the domain name in a normal certificate, or the name after the "*." in a wildcard certificate for a "CERT" application. If the boundary is above the name, the name is allowed.

7. Discussion

The total number of DNS lookups is no more than the number of levels of boundary delegation, plus one if the last boundary doesn't have the NOLOWER keyword. That is unlikely to be more than 2 or 3 in realistic scenarios, and depends on the number of boundaries, not the number of components in the names that are looked up. With shadow records, it will typically be one lookup that matches a shadow record and a second to check below it that gets NXDOMAIN if the shadow record doesn't contain NOLOWER.

Some domains have very irregular boundaries. This may require a relatively large number of records to describe all the boundaries, but it doesn't seem like a number that would challenge modern DNS servers, or need unduly complex scripts to create them. A mechanical translation of the boundary information the Mozilla PSL as of August 2022 creates about 17,000 records.

The wildcard lookup means that each time an application looks up the boundaries for a hostname, the lookup results create DNS cache entries that will not be reused other than for subsequent lookups for the identical hostname. This might cause cache churn, but it seems at worst no more than we already tolerate from DNSBL lookups. If the boundary zone is signed, DNS caches should be able to syntheize some answers from cached wildcards.

8. ABNF syntax of bound records

The syntax of bound records is something like this:

```
; the DNS record contains one string with space separated fields

BOUND = BTAG " " BFLAGS " " BKWDS " " DOMAIN

BTAG = %s"bound=1"

BFLAGS = ( BFLAG *("," BFLAG) ) / "."

BFLAG = s%"NOLOWER" / s%"NOBOUND" / MISCTOK

BKWDS = ( BKWD *("," BKWD) ) / "."

BKWD  = s%"DMARC" / s%"COOKIE" / s%"CERT" / MISCTOK

; miscellaneous tokens for future expansion

MISCTOK = `1*ALPHA
```

9. Security Considerations

The purpose of publishing organization boundaries is to provide advice to third parties that wish to know whether two names are managed by the same organization, allowing those names to be treated "as the same" in some sense. Clients that rely on published boundaries are outsourcing some part of their own security policy to the publisher, so their own security depends on the publisher's boundaries being accurate.

Although in some sense domains are always in control of their subdomains, there are many situations in which parent domains are not expected to influence subdomains. For example, second level domains in global TLDs (gTLDs) operated by registries with contracts with the Internet Corporation for Assigned Names and Numers (ICANN) Since there is no technical bar to a parent publishing records that shadow part or all of the boundary record namespace for delegated subdomains, correct operation depends on the parent and subdomains agreeing about who publishes what.

The DNS is subject to a variety of attacks. DBOUND records are subject to the same ones as any other bit of the DNS, and the same countermeasures, such as using DNSSEC, apply.

10. Variations

Some boundary schemes distinguish between public and private subtrees. If that were useful, a PUBLIC flag keyword could indicate that the subtrees below a boundary were public rather than the default of private.

Since nothing but the boundary records should be published at names with _bound components, one could get the same effect with a new DBOUND RRTYPE, which would avoid the problem of confusion with other TXT wildcards. Its syntax would be similar to a TXT record, a text string, but without the initial tag field. They would still need to be in a separate subtree identified by _bound labels so that the wildcard name coverage would work, so the usual benefits of a unique RRTYPE would not apply.

If third parties want to publish boundary information, they can do it in their own subtree of the DNS. If the boundary information is published by a third party, the client appends the base name of the third party's domain to the name to be looked up. For example, if policy.example were publishing boundary information about boundaries, the records for the test domain described above would be:

```
*._bound.test.policy.exaple IN TXT "bound=1 . . ."
_bound.test.policy.example  IN TXT "bound=1 . . ."
```

The PSL has a little-used wildcard feature where the first label in the name may be "*" to indicate that the boundary is at any name one label deeper than the rest of the name. That is, the asterisk matches a single label, not the usual DNS sense of matching any string of labels.

This feature could be added by allowing a similar * in the domain field of boundary records. For example, if the domain name in a boundary record is "*.example", the client replaces the * with the corresponding element of the domain being matched. If the domain were "www.test.example", the boundary record domain would be treated as though it were "test.example".

11. IANA considerations

IANA is requested to add this entry to the "Underscored and Globally Scoped DNS Node Names" Registry.

RR Type	_NODE NAME	Reference
TXT	_bound	(this document)

Table 1

This document requests that IANA create a registry of dbound Flag keywords. Its registration policy is IETF Review. Its initial contents are as follows. [[NOTE: new flags are likely to change the lookup algorithm]]

Keyword	Reference	Description
NOLOWER	(this document)	No lower level policies
NOBOUND	(this document)	No boundary at this name

Table 2: BOUND Flag Keywords Initial Values

This document requests that IANA create a registry of BOUND Application keywords. Its registration policy is First Come First Served. Its initial contents are as follows. [[Note: New applications don't affect the lookup process, and shouldn't affect existing applications.]]

Value	Reference	Description
. (Any)	(this document)	Any application without a specific boundary record
DMARC	(this document)	DMARC organizational domains
COOKIE	(this document)	HTTP cookies
CERT	(this document)	Owner of certificate requests

Table 3: BOUND Applications Initial Values

12. References

12.1. Normative References

[RFC1034] Mockapetris, P.V., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <https://www.rfc-editor.org/info/rfc4592>.

12.2. Informative References

[PSL] Mozilla Foundation, "Public Suffix List", <https://publicsuffix.org/>.

Appendix A. Implementations

A sample python implementation is available at https://github.com/jrlevine/bound. It includes a library routine to find the boundaries for a domain name, and a script to translate the Mozilla PSL [PSL] into the DNS format. For testing, a copy of the translated PSL is online at bound.services.net.

Appendix B. Change Log

NOTE TO RFC EDITOR: This section may be removed upon publication of this document as an RFC.

06 to -07 Put the four fields into one TXT string. Move third party publishing and the PSL * stuff to a variation. Add few examples.

05 to -06 Editorial changes, add non-goals

04 to -05 Editorial changes, add implementation appendix

03 to -04 Make TXT fields separate strings, add shadow records, update ABND

02 to -03 Add wildcard labels like in the PSL.

01 to -02 Make TXT record the proposal, new RR as alternative.

-00 to -01 Editorial changes to limit standard use to DMARC.

non-WG to -00 Add NOBOUND record to make wildcard matches do the right thing

Rename to match WG name

Author's Address

John Levine
Taughannock Networks

Email: standards@taugh.com
URI: https://jl.ly