Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 21, 2016

J. Levine
Taughannock Networks
November 18, 2015

Publishing Organization Boundaries in the DNS
draft-levine-orgboundary-04

Abstract

Often, the organization that manages a subtree in the DNS is different from the one that manages the tree above it. Rather than describing a particular design, we describe an architecture to publish in the DNS the boundaries between organizations that can be adapted to various policy models.

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 web 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. Some 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.

[[Please direct discussion of this draft to the dbound working group 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:

- o 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.

- o Multiple levels of delegation may be implemented, which is different from irregular boundaries. For example, "ca", "on.ca", and "toronto.on.ca" are irregular boundaries, because they're all handled by the Canadian Internet Registration Authority (CIRA). CentralNIC's "uk.com" would be a second level of delegation below Verisign's com.

- o 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 web cookie setting policies.

In the lookup process below, the boundary point data is stored in the DNS tree in a new BOUND RRTYPE. 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 "foo.bar.abc.example" are not.

Each domain publishes its own policies.

3. RRTYPE format

The BOUND record contains two 16-bit numeric values and an uncompressed domain name. In a master file, they are written as two decimal values and a domain name.

The first numeric value is a bit mask expressing policy options. The only bit currently assigned is 0x0001 (NOLOWER) which means that no lower level boundaries can exist below this one.

The second numeric value is a number identifying the application to which this boundary applies. The number zero is a default for any applications not otherwise specified.

4. Lookup Process

In general, the lookup process takes as input a domain name an application number. 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 defaults, and the DNS root is not within any organization boundary. (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".

The client does a DNS lookup of BOUND records at that name, which will return zero or more BOUND 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.

If a relevant policy record is returned, the domain name in the record is the policy boundary. A policy record is relevant if it its application number is the application’s number, or its application number is zero and there is no record with the application’s number. For example, a check for a boundary above "example.com" would be issued at "example._bound.com", and the expected response could be "BOUND 0 0 com".

If there are no boundaries below the queried point, the policy record contains "BOUND 1 0 ." 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 0 .".

If the relevant record has the NOLOWER bit 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:

o When evaluating "www.foo.example.com", the first query would be to "www.foo.example._bound.com". If the reply to this is "BOUND 0 0 com", then the second query would go to "www.foo._bound.example.com".

o When evaluating "www.example.on.ca", the first query would be to "www.example.on._bound.ca". If the reply to this is "BOUND 0 0 on.ca", the next lookup would be to "www._bound.example.on.ca".

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

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 BOUND 0 0 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 BOUND 0 0 .
  _bound.test   IN BOUND 0 0 .
```

In domains with irregular policy boundaries, multiple records in the record describe the boundary points. For example, in the CA (Canada) TLD, for national organizations there might be a boundary directly below the national TLD; for provincial organizations there might be a boundary below a provincial subdomain such as "on.ca"; and for local (e.g., municipal) organizations, a boundary below a municipal subdomain such as "toronto.on.ca" might exist. A suitable set of of records covers this structure. The closest encloser rule in RFC 4592 [RFC4592] makes the wildcards match the appropriate names.

```
*._bound.ca            IN BOUND 0 0 ca
*.on._bound.ca         IN BOUND 0 0 on.ca
*.toronto.on._bound.ca IN BOUND 0 0 toronto.on.ca"
```

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 DBOUND 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.

6. Application scenearios

Here are some ways that applications might use BOUND data.

6.1. 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 the "cookie" application for both the request's domain and the cookie domain. If they are not separated by a boundary, the request is allowed.

6.2. SSL Certificates

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

6.3. 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.)

7. Discussion

The total number of DNS lookups is the number of levels of boundary delegation, plus one if the last boundary doesn't have the NOLOWER flag. 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.

Some domains have very irregular boundaries. This may require a large number of records to describe all the boundaries, perhaps several hundred, but it doesn’t seem like a number that would challenge modern DNS servers.

The wildcard lookup means that each time an application looks up the boundaries for a hostname, the lookup results use 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.

8. 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, the Internet Corporation for Assigned Names and Numers (ICANN) contracted global TLDs (gTLDs) and registers second level domains. 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 responses, such as using DNSSEC, apply.

9. Variations

Since nothing but BOUND records should be published at names with _bound components, one could get the same effect with TXT records, e.g.:

```
  *.toronto.on._ob.ca IN TXT "bound=1 0 0 toronto.on.ca"
```

The "bound=1" tag is to prevent confusion when a domain publishes a wildcard such as *.example.com that could match a _bound name.

If third parties wanted to publish boundary information, they could do it in their own subtree of the DNS. For example, if polgroup.example were publishing boundary information about boundaries, the records for the test domain described above would be:

```
*._bound.test.polgroup.exaple IN BOUND 0 0 .
_bound.test.polgroup.example  IN BOUND 0 0 .
```

10. IANA considerations

This document defines a new DBOUND RRTYPE. IANA has assigned value TBD.

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

Bit	Reference	Description
0x0001 (NOLOWER)	(this document)	No lower level policies

Table 1: BOUND Flag Bits Initial Values

This document requests that IANA create a registry of BOUND Applications. 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
0 (Any)	(this document)	Any application
1 (Cookie)	(this document)	HTTP web cookies
2 (Cert)	(this document)	SSL certificate authorities
3 (DMARC)	(this document)	DMARC organizational domains

Table 2: BOUND Applications Initial Values

11. References

11.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://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, <http://www.rfc-editor.org/info/rfc4592>.

11.2. Informative References

[PSL] Mozilla Foundation, "Public Suffix List", Nov 2015.

Appendix A. Change Log

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

A.1. Changes from -03 to -04

Editorial changes, fix speling errors.

A.2. Changes from -02 to -03

New BOUND record type and modified lookup procedure.

A.3. Changes from -01 to -02

Add ABNF.

MSK overhaul of the middle part.

Put the wildcards back.

A.4. Changes from -00 to -01

Take out wildcards and put everything in one record.

Add DNS nits.

Author’s Address

John Levine
Taughannock Networks
PO Box 727
Trumansburg, NY 14886

Phone: +1 831 480 2300
Email: standards@taugh.com
URI: http://jl.ly

Network Working Group Y. Pettersen
Internet-Draft Vivaldi Technologies AS
Updates: 6265 (if approved) February 13, 2014
Intended status: Experimental
Expires: August 17, 2014

The Public Suffix Structure file format and its use for Cookie domain validation
draft-pettersen-subtld-structure-10

Abstract

This document defines the term "Public Suffix domain" as meaning a domain under which multiple parties that are unaffiliated with the owner of the Public Suffix domain may register subdomains. Examples of Public Suffix domains include "org", "co.uk", "k12.wa.us" and "uk.com". It also defines a file format that can be used to distribute information about such Public Suffix domains to relying parties. As an example, this information is then used to limit which domains an Internet service can set HTTP cookies for, strengthening the rules already defined by the cookie specification. This specification updates RFC 6265 [RFC6265] by defining the term "Public Suffix domain".

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

1. Introduction

The Domain Name System (DNS) [RFC1034] used to name Internet hosts allows a wide range of hierarchical names to be used to indicate what a kind of business a given host is engaged in. Some are implemented by the owners of a domain, such as by creating subdomains for certain tasks or functions, while others, called Public Suffixes (or registry-like domains), are created by the Top Level Domain (TLD) registry owner or individual domain owners to indicate what kind of service the hosts under the domain provides, e.g., commercial, educational, governmental or geographical location, such as city or state.

While this system makes it relatively easy for TLD administrators to organize online services, and for the user to locate and recognize relevant services, this flexibility causes various security and privacy-related problems when services located at different hosts are allowed to share data through functionality administrated by the client, e.g., HTTP state management cookies [RFC6265] and cross-document information sharing in ECMAScript DOM. Most of these information-sharing mechanisms make the process of sharing easy, perhaps too easy, since, in many cases, there is no mechanism to ensure that the servers receiving the information really want it. It is also often difficult to determine the original source of the information being shared. To some extent, [RFC2965] tried to address some of these concerns for cookies, in that clients that send [RFC2965]-style cookies also send the target domain for the cookie along with the cookie so that the recipient can verify that the cookie has the correct domain. Unfortunately, [RFC2965] was never widely deployed in clients or on servers. The recipient server(s) can make inappropriate information sharing more detectable by requiring the information to contain data identifying the source, as well as by assuring the integrity of the data, e.g., by using cryptographic technologies. However, these techniques tend to be computationally costly.

There are two problem areas:

- Incorrect sharing of information between non-associated services e.g., example1.com and example2.com or example1.co.uk and example2.co.uk -- That is, the information may be distributed to all services within a given Top Level Domain or Public Suffix domain. Sharing within a TLD is usually prevented by a simple rule that does not permit it, but that is more difficult for general Public Suffix domains, since they do not have a well defined pattern.

- Undesirable information sharing within a single service -- This is, in particular, a problem for services that sell hosting services to many different customers, such as web hotels, where the service itself has little or no control of the customers' actions.

While both these problems are in some ways similar, they call for different solutions. This specification will only propose a solution for the first problem area. The second problem area must be handled separately. This specification will first define what Public Suffixes are; then it will propose a file format that can be used to distribute information about the Public Suffixes within a Top Level Domain, e.g., that the TLD have several Public Suffix domains, such as co.tld, ac.tld, org.tld. Finally it will show how this information can be used to determine when information sharing through cookies is not desirable.

2. Public Suffix domains

A Public Suffix domain is used very much like a Top Level Domain. The owner or operator of the domain allow unaffiliated third parties to register domain names in the Public Suffix domain and to control all activity inside the registered domain.

While most such domains are open to registration by the general public, the owner of the Public Suffix domain may have defined restrictions on which third parties may register a domain, such as only private persons, schools, or government agencies, to mention a few. Such limitations do not change the fact that each domain registrant is nominally independent of all other domain registrants in the Public Suffix domain, as well as of the owner of the Public Suffix domain.

Just like a domain under a TLD may be a Public Suffix domain, a domain registered under a Public Suffix domain may also be a Public Suffix domain. Examples of this are the state.us and city.state.us Public Suffix domains in the dot-US ccTLD.

There are various categories of Public Suffix domains. The most common category is the second-level domain, used by many ccTLDs to group content, such as co.tld for commercial, ac.tld for academic institution, and gov.tld for government. Another common category is Public Suffixes dedicated to geographical locations, such as as states, provinces, and cities, such as the city.state.us domain organization used by the US ccTLD, possibly with more Public Suffix domains within these domains. A third category is ISP shared hosting, and "vanity" domain names, e.g., country.com. In addition, a number of social websites provide their users with direct access names under their domains (e.g., http://example-user.example.com, rather than using http://www.example.com/example-user URLs).

Information about Public Suffix domains can be used in several security features in clients:

- o Blocking websites’ ability to set cookies to the Public Suffix domain

- o Limiting the ability of active content, such as EcmaScript, from affecting content in independent domains that happen to share the same Public Suffix domain as the source domain

- o Highlighting the actual domain in the displayed URL in the client’s UI, to reduce the potential of a malicious site misleading the user with a URL that identifies the host as www .well-known-site.com.example.pubsuffix-domain.tld, which might lead users to think they are visiting www.well-known-site.com rather than a site in the domain example.pubsuffix-domain.tld

As there is currently no reliable method in DNS or other protocols that allows clients to automatically recognize a Public Suffix domain, the owner of such a domain must self-declare the domain’s status as a Public Suffix domain and register it with each of the repositories that track such information. Such self-registration may lead to inconsistencies between the various repositories, which could cause security problems to develop. A more reliable method might be that the TLD registrar collect such information from their registered domains, and make it available to relying parties. Section 3 presents a XML-based format for how this information can be published and shared by the TLD registrar.

3. The Public Suffix Structure file format

The Public Suffix Structure file format specifies how to encode information about Public Suffix domains inside a TLD. It is based on XML and is able to specify Public Suffixes and exceptions to any level of the domain hierarchy that is desirable.

3.1. Domain list format

The domain list file can contain a list of subdomains that are considered Public Suffix domains, as well as a special list of names that are not top level domains. None of the domain lists need specify the TLD name, since that is implied either by the file that is parsed or by the content of the <tld> tag. The domain names listed MUST be encoded in punycode, as specified by [RFC5891].

3.1.1. Domain list schema

The domain list is an XML file that follows the following schema:

```
default namespace = "http://xmlns.opera.com/tlds"

start =
    element tld {
      attribute levels { xsd:nonNegativeInteger | "all"},
      attribute name { xsd:NCName },
      (domain | registry)*
    }
registry =
    element registry {
      attribute levels { xsd:nonNegativeInteger },
      attribute name { xsd:NCName },
      attribute all { string "true" | string "false" },
      (domain | registry)*
    }
domain =
    element domain {
      attribute name { xsd:NCName }
    }
```

The domain list file usually contains a single <tld>-block (but may contain multiple entries), which may contain multiple registry and domain blocks, and a registry block, which define a Public Suffix domain, may also contain multiple registry and domain blocks. When used alone, the <tld>-block MAY contain a name field identifying the TLD name; if there are multiple <tld>-blocks, each block MUST specify the name field.

Both <domain> and <registry> tags MUST contain a name attribute identifying the domain or registry. The <tld>-block MAY have a name attribute, but in files with a single <tld>-block this name MUST be ignored by clients, which must instead use the name of the TLD used to request the file.

All names SHOULD be punycode encoded [RFC5891] to make it possible for clients unaware of either Unicode or IDNA to use the document.

The <tld>- and <registry>-blocks MAY contain an attribute, "levels", specifying how many levels below the current domain are Public Suffixes. The default is "none", meaning that the default inside the current domain level is that labels are ordinary domains and not Public Suffix domains. If the value of the "levels" attribute is 1 (one) by default all next-level labels within the registry/TLD are Public Suffix domains, not normal domains. If the value of the "levels" attribute is the case-insensitive token "all", then all subdomains domains below the current domain are Public Suffix domains, by default.

A <registry>-block with the attribute "all" set to "true" inside the declaration for the registry domain example.tld indicates that all domains x.example.tld are also Public Suffix domains, by default, unless a domain is specified differently by a different declaration. The registry-all block may contain additional <registry>- or <domain>-blocks, which then apply to domains foo.x.example.tld, for all domains x, except those that have separate entries. This allows specification of wildcard structures, where the structure for lower domains are similar for all domains.

Implementations MUST ignore attributes and syntax they do not recognize.

3.1.2. Domainlist interpretation

For each new <registry>- or <domain>-block within the <tld>- or <registry>-block, the effective domain name to which the block applies is the name of the block prepended to the ".parentdomain" of the effective domain name of the containing block.

For the <tld>-block the effective domain name is the name of the TLD the client is evaluating, and for the <registry>-block named "example" the effective name becomes example.tld.

```
<?xml version="1.0" encoding="UTF-8"?>
<tld xmlns="http://xmlns.opera.com/tlds" name="tld" levels="1" >
    <registry name="co" levels="0">
      <registry name="state" />
    </registry>
    <registry name="province">
      <registry all level="1">

        <domain name="school" />
      </registry>
    </registry>
    <registry name="example" levels="1" />
    <domain name="parliament" />
</tld>
```

In the above example, the specification is for the TLD "tld". By default any second level domain "x.tld" is a Public Suffix domain; although, parliament.tld is not a Public Suffix domain, but a normal domain.

In the example TLD, however, the co.tld registry has a sub registry "state.co.tld", while all other domains in the co.tld domains are ordinary domains.

Additionally, all domains x.province.tld are Public Suffixes, and all school.x.province.tld are normal domains for all domains x in province.tld.

Also, the registry example.tld has defined all domains y.example.tld as Public Suffixes, with no exceptions.

3.2. Public Suffix Structure as a web service

The Public Suffix structure file can be provided as an HTTP service, managed by either the application vendor, the TLD owners, or some other trusted organization, and it can be located at a URI location that, when queried, returns information about a TLD’s domain structure. The client can then use this information to decide what actions are permitted for the protocol data the client is processing. The procedure for use as a service is as follows:

o The client retrieves the domain list for the Top Level Domain "tld" from the vendor specified URI https://tld-structure.example.com/tld/domainlist . Multiple alternative URIs for a fallback procedure may be specified.

- The Content-Type of the returned list MUST be application/subdomain-structure.
- The retrieved specification SHOULD be cached by the client for at least 30 days.
- The TLD owner SHOULD update the list at least 90 days before a new sub-domain becomes active.
- If no specification can be retrieved the user agent MAY fall back to alternative, undefined methods, depending on the profile.

3.3. Securing the domain information

Individuals with malicious intent may wish to modify the domain list served by the service location to either classify a domain incorrectly as a Public Suffix domain or to hide a Public Suffix domain’s classification. Besides obviously securing the hosting locations, this also means that the content served will have to be secured.

1. Digitally sign the specification, using one of the available message signature methods, e.g., S/MIME [RFC2311]. This will secure the content during storage both at the client and the server, as well as during transit. The drawback is that the client must implement decoding and verification of the message format that it may not already support, which may be problematic for clients having limited resources.
2. Use an encrypted connection, such as HTTP over TLS [RFC2818], which is supported by many clients already. Unfortunately, this method does not protect the content when stored by the client.
3. Use XML Signatures [RFC3275] to create a signature over the specification. This method is currently not defined.

This specification recommends using HTTP over TLS, and the client MUST use the non-anonymous cipher suites, to secure the transport of the specification. The client MUST ensure that the hostname in the certificate matches the hostname used in the request.

4. A Public Suffix Structure file format profile for HTTP Cookies

The HTTP State management cookies area is one where it is important, both for security and privacy reasons, to ensure that unauthorized services cannot set cookies for another service. Inappropriate cookies can affect the functionality of a service, but they may also be used to track the users across services in an undesirable fashion. Neither the original Netscape cookie specification[NETSC], [RFC2965] or [RFC6265] are adequate in many cases.

The original Netscape specification’s rules required only that the target domain must have one internal dot (e.g., example.com) if the TLD belongs to a list of generic TLDs (gTLD), while for all other TLDs the domain must contain two internal dots (e.g., example.co.uk). The latter rule was never properly implemented, in particular due to the many flat ccTLD domain structures that are in use. (The successor [RFC6265] has since expanded this policy to exclude domains listed in client-specific lists of "public suffixes").[RFC2965] set the requirement that cookies can only be set for the server’s parent domain.

Unfortunately, both the [NETSC] and [RFC2965] policies still left open the possibility of setting cookies for a Public Suffix domain by setting the cookie from a host name example.pubsuf.tld to the domain pubsuf.tld, which is by itself legal, but not desirable, because that means that the cookie can be sent to numerous websites either revealing sensitive information, or interfering with those other websites without authorization. As can be seen, these rules do not work satisfactorily, especially when applied to ccTLDs, which may have a flat domain structure similar to the one used by the generic .com TLD, a hierarchical Public Suffix domain structure like the one used by the .uk ccTLD (e.g., .co.uk), or a combination of both. However, there are also gTLDs, such as .name, for which cookies should not be allowed for the second-level domains, as these are generally family names shared between many different users, not service names. A partially effective method for distinguishing service names from Public Suffix domains by using DNS was developed by Opera Software ASA. However, this method was not immune to TLD registries that use Public Suffix domains as directories or to services that do not define an IP address for the domain name. Using the Public Suffix Structure file format to retrieve a list of all Public Suffix domains in a given TLD will solve both those problems.

4.1. Procedure for using the Public Suffix Structure file format for cookies

When receiving a cookie, the client must first perform all the checks required by the relevant specification. Upon completion of these checks the client then performs the following additional verification checks if the cookie is being set for the server's parent, grand-parent domain (or higher):

1. If the Public Suffix domain structure of the TLD is not known already, or the structure information has expired, according to the client's policies, the client should retrieve or revalidate the structure specification from the server hosting the specification, according to Section 3. If retrieval is unsuccessful, and no copy of the specification is known, the client MAY use alternative information or heuristics to decide the domain's status. Upon successful retrieval the specification is evaluated as specified in Section 3. If the target domain is designated as a Public Suffix domain, then the cookie MUST be discarded or, alternatively, processed as if it had not specified a domain attribute.

2. If the target domain is not a Public Suffix domain, the cookie is accepted (unless other policies configured for the client prevent this).

4.2. Third party cookies

Use of HTTP Cookies, combined with HTTP requests to resources that are located in domains other than the one the user actually wants to visit, have caused widespread privacy concerns. The reason is that multiple websites can link to the same independent website, e.g., an advertiser, who may then use cookies to build a profile of the visitor, which can be used to select advertisements that might be of interest to the user.

Some clients have therefore implemented restrictions on what cookie related activities are accepted in relation to a third-party domain. Frequently, such restrictions are based on determining whether the two hosts share the same immediate parent domain as a domain suffix, or if the first domain of the two is a parent domain (suffix) of the other.

This determination method might incorrectly classify a third-party server as a first party if the immediate parent domain of the first party server is a Public Suffix domain, and possibly break the user's privacy expectations.

To avoid such misclassifications, when the two servers have the first party server’s immediate parent domain as a shared suffix, clients SHOULD apply the procedure specified in Section 4.1 for this domain, and, if this domain is determined to be a Public Suffix domain, the second host must be considered a third party. That is, if the first party server example.co.uk causes a request for a resource at example3.co.uk the parent domain co.uk is determined to be a Public Suffix domain, and example3.co.uk is therefore a third-party server, even if they share the same immediate parent domain.

5. Examples

The following examples demonstrate how the Public Suffix Structure file format can be used to decide cookie domain permissions.

5.1. Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<tld xmlns="http://xmlns.opera.com/tlds" name="tld" levels="1" >
    <domain name="example" />
</tld>
```

This specification means that all names at the top level are Public Suffix domains, except "example.tld" for which cookies are allowed. Cookies are also implicitly allowed for any y.x.tld domains.

5.2. Example 2

```
<?xml version="1.0" encoding="UTF-8"? >
<tld xmlns="http://xmlns.opera.com/tlds" name="tld" >
    <registry name="example1" levels="1" />
    <registry name="example2" levels="1" />
</tld>
```

This specification means that example1.tld and example2.tld and any domains (foo.example1.tld and bar.example2.tld) are Public Suffix domains for which cookies are not allowed; for any other domains cookies are allowed.

5.3. Example 3

```
<?xml version="1.0" encoding="UTF-8"?>
<tld xmlns="http://xmlns.opera.com/tlds" name="tld" >
    <registry name="example1" levels="1" />
    <registry name="example2" levels="1" >
       <domain name="example3" />
    </registry>
</tld>
```

This example has the same meaning as Example 2, but with the exception that the domain example3.example2.tld is a regular domain for which cookies are allowed.

6. IANA Considerations

This specification also requires that responses are served with a specific media type. Below is the registration information for this media type.

6.1. Registration of the application/subdomain-structure Media Type

Type name : application

Subtype name: subdomain-structure

Required parameters: none

Optional parameters: none

Encoding considerations: The content of this media type is always transmitted in binary form.

Security considerations: See Section 7.

Interoperability considerations: none

Published specification: This document

Additional information:

Magic number(s): none

File extension(s):

Macintosh file type code(s):

Person & email address to contact for further information: Yngve N. Pettersen

Email: yngve@vivaldi.com

Intended usage: common

Restrictions on usage: none

Author/Change controller: Yngve N. Pettersen

Email: yngve@vivaldi.com

7. Security Considerations

Retrieval of the Public Suffix Structure specifications is vulnerable to denial of service attacks or loss of network connection. Hosting the specifications at a single location can increase this vulnerability, although the exposure can be reduced by using mirrors with the same name but hosted at different network locations. This protocol is as vulnerable to DNS security problems as any other [RFC2616] HTTP-based service. Requiring the specifications to be digitally signed or transmitted over a authenticated TLS connection reduces this vulnerability.

Section 4 of this document describes using the domain list defined in Section 3 as a method of increasing security and privacy. The effectiveness of the domain list for this purpose, and the resulting security and privacy improvements for the user, depend both on the integrity of the list, and its correctness. The integrity of the list depends on how securely it is stored in the repository and how securely it is transmitted. This specification recommends downloading the domain list using HTTP over TLS [RFC2818], which makes the transmission as secure as the message authentication mechanism used (encryption is not required), and the servers should be configured to use the strongest available key lengths and authentication mechanisms. An alternative or complimentary approach would be to digitally sign the files.

The correctness of the list depends on how well the TLD registry defined it, or how well the list maintainer have been able to collect correct information. A list that does not include some Public Suffix domains may expose the client to potential privacy and security problems, but the situation would not be any worse than it would be without this protocol and profile, while a subdomain incorrectly classified as a Public Suffix domain can lead to denial of service for the affected services. Both of the problems can be prevented by careful construction and auditing of the lists, both by the TLD regis try and by interested third parties.

8. Acknowledgments

Anne van Kesteren assisted with defining the XML format in Section 3.1.1.

The Public Suffix List project [PUBSUFFIX] was initiated by members of the Mozilla Community, and members of the project, in particular Gervase Markham, have provided input to this document.

9. References

9.1. Normative References

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

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

[RFC2311] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3275] Eastlake, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011.

9.2. Non-normative references

[NETSC] "Persistent Client State HTTP Cookies", <http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.3/reference/cookies.html>.

[PUBSUFFIX]
"The Homepage of the Public Suffix List, a list of registry-like domains gathered by volunteers.", <http://publicsuffix.org/>.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

Appendix A. Collection of information for the TLD structure specification

This document does not define how the information encoded in the TLD Structure Specification is gathered.

There are several methods available for collecting the information encoded in the TLD Structure Specification, the two main ones being:

Data provided by the TLD registry owner through a machine readable repository at well known locations

Data gathered by one or more application vendors based on publicly available information, such as the Mozilla Project’s Public Suffix List[PUBSUFFIX],

Appendix B. Alternative solutions

A possible alternative to the format specified in Section 3, encoding the information directly in the DNS records for the Public Suffix domain, using a DNS extension.

Accessing this type of information requires that the client or its environment is able to directly access the DNS network. In many environments, e.g., firewalled systems, this may not be possible. Also, not all runtime environments can provide this information, which may lead to a DNS client embedded directly in the client.

For some applications, it may be necessary, due to system limitations, to access this information through an online web service in order to provide the necessary information for each hostname or domain visited. A web service may, however, introduce unnecessary privacy problems, as well as delays each time a new domain is tested.

Appendix C. Open issues

o Download location URI for the original domain lists

o Should Digital signatures be used on the files, instead of using TLS?

Author's Address

Yngve N. Pettersen
Vivaldi Technologies AS
Norway

Email: yngve@vivaldi.com

IETF   A. Sullivan
Internet-Draft   Dyn, Inc.
Intended status: Standards Track   J. Hodges
Expires: August 21, 2016   PayPal
                                                       February 18, 2016

Asserting DNS Administrative Boundaries Within DNS Zones
draft-sullivan-domain-policy-authority-02

Abstract

Some entities on the Internet make inferences about the administrative relationships among Internet services based on the domain names at which those services are offered. At present, it is not possible to ascertain organizational administrative boundaries in the DNS; therefore such inferences can be erroneous. Mitigation strategies deployed so far will not scale. This memo provides a means to make explicit assertions regarding certain kinds of administrative relationships between domain names.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Motivation

Many Internet resources and services, especially at the application layer, are identified primarily by domain names [RFC1034]. As a result, domain names have become fundamental elements in building security policies and also in affecting user agent behaviour. Discussion of several of these uses, and some of the associated issues can be found in [I-D.sullivan-dbound-problem-statement].

Historically, attempts to build the security policies have relied on the public suffix list (see discussion in [I-D.sullivan-dbound-problem-statement]). We proceed from the view that some uses of the public-suffix list never were going to achieve their goal, and that the public/private distinction may be a poor proxy for the kinds of relationships that are actually needed. At the same time, it will be necessary to continue to use something like a public suffix list for some important classes of behaviour (both to achieve acceptable performance characteristics and to deal with deployed software). Therefore, the proposal below does not attempt to address all the issues in [I-D.sullivan-dbound-problem-statement], but offers a way to solve one important class of problems -- the "orphan type" policies.

1.1. Organization of This Memo

[[CREF1: I find this section awkward here. Ditch it? --ajs@anvilwalrusden.com]]

Necessary terminology is established in Section 2. Section 3 provides an overview of what the mechanism is supposed to do. Then, Section 4 discusses the conditions where the technique outlined here may be useful, and notes some cases that the technique is not intended solve. A definition of a new RRTYPE to support the technique is in Section 5. There is some discussion of the use of the RRTYPE in Section 6. Section 7 attempts to show how the mechanism is generally useful. Then, Section 8 offers an example portion of a DNS tree in an effort to illustrate how the mechanism can be useful in certain example scenarios. Section 9 notes some limitations of the mechanism. Section 10 outlines how the mechanism might be used securely, and Section 11 addresses the internationalization consequences of the SOPA record. Finally, Section 12 includes the requests to IANA for registration.

2. Terminology

The reader is assumed to be familiar with the DNS ([RFC1034] [RFC1035]) and the Domain Name System Security Extensions (DNSSEC)

([RFC4033] [RFC4034] [RFC4035] [RFC5155]). A number of DNS terms can be found in [RFC7719].

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

The terms "policy realm" and "policy authority" are defined in [I-D.sullivan-dbound-problem-statement]. For the purposes of discussion here, it is important to remember that it is a matter of fact as to whether two domains lie in the same policy realm. The point of the mechanism here is not to create such facts, but merely to expose them. The terms "inheritance type" and "orphan type" are also defined in [I-D.sullivan-dbound-problem-statement]. The text below attempts to apply the categories when they seem useful.

3. Overview of Start Of Policy Authority (SOPA)

When an application is attempting to make security decisions based on domain names, it needs to answer questions about the relation between those names. Suppose that the question to be answered is, "Given any two domain names, do they lie in the same policy realm appropriate for a given application?" In order to answer this, there are two pieces of information needed: first, does the application need an inheritance or orphan type of policy? Second do the two names lie in the same policy realm? For orphan types of policy, the best way to determine whether two names lie in the same policy realm is to look for assertions about the two domain names. A good place to look for assertions about domain names is in the DNS.

This memo presents a way to assert that two domains lie in the same policy realm by placing a resource record (RR) at the affected domain names in the DNS. The mechanism requires a new resource record type (RRTYPE). It is called SOPA, for "Start Of Policy Authority" and echoing the Start Of Authority or SOA record. While there are reported difficulties in deploying new RRTYPEs, the only RRTYPE that could be used to express all the necessary variables is the TXT record, and it is unsuitable because it can also be used for other purposes (so it needs to be covered itself). The use of this mechanism does not require "underscore labels" to scope the interpretation of the RR, in order to make it possible to use the mechanism where the underscore label convention is already in use. The SOPA RRTYPE is class-independent.

The use of SOPA records can do one of two things: it can confirm that two names are in the same policy realm, or it can refute a claim that they are. In order to learn whether a.long.example.com and b.example.com are in the same policy realm, perform a DNS query for the SOPA record for a.long.example.com. If the answer's RDATA contains b.example.com, that is an assertion from the nameservers for a.long.example.com that it is in the same policy realm as b.example.com. Next, make a DNS query for the SOPA record for b.example.com. If the answer's RDATA contains a.long.example.com, then the two names are in the same policy realm. A positive policy realm relationship ought to be symmetric: if example.com is in the same policy realm as example.net, then example.net should be (it would seem) in the same policy realm as example.com. In principle, then, if a SOPA RR at a.long.example.com provides a target at b.example.com, there should be a complementary SOPA RR at b.example.com with a target of a.long.example.com. Because of the distributed nature of the DNS, and because other DNS administrative divisions need not be congruent to policy realms, the only way to know whether two domain names are in the same policy realm is to query at each domain name, and to correlate the responses. If any of the forgoing conditions fails, then the two names are not in the same policy realm.

[[CREF2: Something that could be useful here is a transitivity bit in the SOPA record. That would allow SOPAs between a.example.com and example.com, and b.example.com and example.com, to mean that a.example.com and b.example.com are also in the same realm (but you could shut it off by clearing the bit). I'm leery of this because of the potential for abuse and also because I doubt it saves very much. Might be useful for administrative saving, but it won't save lookups. --ajs@anvilwalrusden.com]]

It is also possible for a SOPA record to contain the explicit statement that other names do not lie in the same policy authority as it. This negative assertion permits processing to stop. If the assertion is about all other names, then the capability is functionally equivalent to declaring a name to be a public suffix.

In operation where latency is an important consideration (such as in a web browser), it is anticipated that the above correlations could happen in advance of the user connection (that is, roughly the way the existing public suffix list is compiled), and then additional queries could be undertaken opportunistically. This would allow the detection of changes in operational policy and make maintenance of the installed list somewhat easier, but not require additional DNS lookups while a user is waiting for interaction.

While many policies of the sort discussed in [I-D.sullivan-dbound-problem-statement] appear to be based on domain names, they are actually often only partly based on them. Often, there are implicit rules that stem from associated components of composite names such as URIs [RFC3986], e.g., the destination port

[RFC6335] or URI scheme [RFC4395] (or both). It is possible to make those assumptions explicit, but at the cost of expressing in the resulting resource record a tighter relationship between the DNS and the services offered at domain names. SRV [RFC2782] records offer a mechanism for expressing such relationships, and a SOPA record in conjunction with an SRV record appears to provide the necessary mechanism to express such relationships. (SRV records use underscore labels, and this is an example of why underscore labels themselves need to be coverable by SOPA records.)

3.1. Identifying a Target Name for Policy Authority

The RDATA of a SOPA RR contains a "target name" that either lies in the same policy realm as the owner name of the RR, or that lies outside of that policy realm. The SOPA record is therefore an assertion, on the part of the authoritative DNS server for the given owner name, that there is some policy relationship between the owner name and the target name. If a given owner name lies in the same policy realm as several other target names, an additional RR is necessary for each such relationship, with one exception. It is not uncommon for a name to have policy relationships with all the children beneath it. Using the SOPA RR, it is possible to specify that the policy target is all the names beneath a given owner name, by using a wildcard target.

4. Use Cases

In the most general sense, this memo presents a mechanism that can be used either as a replacement of the public suffix list <publicsuffix.org>, or else as a way to build and maintain such a list. Performance characteristics may make the mechanism impractical as a full replacement, in which case a list will likely need to be built and maintained. In the latter case, this mechanism is still preferable because it aligns the policy assertions with the operation of the domains themselves, and allows maintenance to be distributed in much the way the operation of the DNS is (instead of being centralized).

It is worth noting that the mechanism outlined here could be used for names that are not along the same branch of the DNS tree (i.e. it could permit the statement that the policy authority of some.example.com and some.other.example.net is the same). Such uses are unlikely to work in practice and probably should not be used for general purposes. Most deployed code implicitly uses ancestor-descendent relations as part of understanding the policy, and such code will undoubtedly ignore cross-tree dependencies. [[CREF3: This relaxes a restriction that was in previous versions, which officially specified the use only for ancestor-descendent uses. It seems better to make that a deployment consideration so that the restriction could be relaxed in some circumstances where it would be appropriate. --ajs@anvilwalrusden.com]]

By and large, the mechanism is best suited to "orphan" types of policy. Where inheritance types of policy can use this, it is mostly by treating the mechanism as a generator for public suffix boundaries.

4.1. Where SOPA Works Well

HTTP state management cookies The mechanism can be used to determine the scope for data sharing of HTTP state management cookies [RFC6265]. Using this mechanism, it is possible to determine whether a service at one name may be permitted to set a cookie for a service at a different name. (Other protocols use cookies, too, and those approaches could benefit similarly.) Because handling of state management cookies often happens during user interaction, this use case probably requires a cached copy of the relevant list. In that case, the mechanism can be used to maintain the list.

User interface indicators User interfaces sometimes attempt to indicate the "real" domain name in a given domain name. A common use is to highlight the portion of the domain name believed to be the "real" name -- usually the rightmost three or four labels in a domain name string. This has similar performance needs as HTTP state management cookies.

Setting the document.domain property The DOM same-origin policy might be helped by being able to identify a common policy realm. This case again has a need for speedy determination of the appropriate policy and would benefit from a cached list. It is likely that the SOPA record on its own is inadequate for this case, but the combination of SOPA and SRV records might be helpful.

SSL and TLS certificates Certificate authorities need to be able to discover delegation-centric domains in order to avoid issuance of certificates at or above those domains. More generally, a CA needs to decide whether, given a request, it should sign a particular domain. This can be especially tricky in the case of wildcards.

HSTS and Public Key Pinning with includeSubDomains flag set
Clients that are using HSTS and public key pinning using includeSubDomains need to be able to determine whether a subdomain

is properly within the policy realm of the parent. An application performing this operation must answer the question, "Should I accept the rules for using X as valid for Y.X?" This use case sounds like an inheritance type, but it is in fact an orphan type.

Linking domains together for reporting purposes It can be useful when preparing reports to be able to count different domains as "the same thing". This is an example where special use of SOPA even across the DNS tree could be helpful.

4.2. Where SOPA Works Less Well

Email authentication mechanisms Mail authentication mechanisms such as DMARC [RFC7489] need to be able to find policy documents for a domain name given a subdomain. This use case is an inheritance type. Because the point of mechanisms like DMARC is to prevent abuse, it is not possible to rely on the candidate owner name to report accurately its policy relationships. But some ancestor is possibly willing to make assertions about the policy under which that ancestor permits names in the name space. This sort of case can only use SOPA indirectly, via a static list that is composed over time by SOPA queries. Other mechanisms will likely better satisfy this need.

5. The SOPA Resource Record

The SOPA resource record, type number [TBD1], contains two fields in its RDATA:

Relation: A one-octet field used to indicate the relationship between the owner name and the target.

Target: A field used to contain a fully-qualified domain name that is in some relationship with the owner name. This field is a maximum of 255 octets long, to match the possible length of a fully-qualified domain name.

```
                      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Relation    |                                               /
 +-+-+-+-+-+-+-+-+                                               /
 /                             Target                            /
 /                                                               /
 /                                                               /
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

5.1. The Relation Field

The relation field is REQUIRED and contains an indicator of the relationship between the owner name and the target name. This memo specifies two possible values:

Value	Setting	Meaning
0	Excluded	The target is not in the same policy realm as the owner name
1	Included	The target is in the same policy realm as the owner name

Table 1

Additional values may be defined in future, according to the rules set out in Section 12.

5.2. The Target Field

The target field contains a fully-qualified domain name, and is REQUIRED to be populated. The name MUST be a domain name according to the rules in [RFC1034] and [RFC1035], except that the any label of the target MAY be the wildcard character ("*"; further discussion of wildcards is in Section 6.4). The target MUST be sent in uncompressed form [RFC1035], [RFC3597]. The target MUST NOT be an alias [RFC2181], such as the owner name of a CNAME RR [RFC1034], DNAME RR [RFC6672], or other similar such resource records. Note that this is a fully-qualified domain name, so the trailing null label is required. [[CREF4: This is a change from previous versions; previously, the target was a root-relative domain name. So it’s now example.com. and used to be example.com (no trailing dot) when in presentation format. The new form makes this a domain name, whereas before it could really have been a text field. Not sure which is better. --ajs@anvilwalrusden.com]]

The target name SHOULD be either an ancestor, a descendent, or a sibling of the owner name in the record. This requirement is intended to limit the applicability of the SOPA RR to names in the same DNS hierarchy, thereby avoiding possible negative side effects of unbounded linkages across disparate DNS subtrees, including those subtrees rooted close to, or immediately below, the DNS root. In special uses, however, it may be desirable to link across the DNS tree. General-purpose clients MAY ignore target names that are neither an ancestor, nor a descendent, nor a sibling of the owner name in the record (and abort processing) in order to avoid the aforementioned negative side-effects.

Targets MAY contain any series of octets, in order to accommodate labels other than LDH labels [RFC6365]. No processing of labels prior to matching targets is to be expected, however, and therefore internationalized domain name targets use whatever form they appear in the DNS. In particular, IDNA labels [RFC5890], [RFC5891], [RFC5892], [RFC5893], [RFC5894] SHOULD appear in A-label form. A SOPA-using client that receives a target containing octets outside LDH MUST NOT treat the affected labels as U-labels, because there is no way to discover whether the affected label is encoded as UTF-8 or something else.

6. Expressing Different Policies with the SOPA RRTYPE

A SOPA RR has one of three different functions. The first is to claim that two domain names are not in the same policy realm ("exclusion"). The second is to claim that two domain names are in the same policy realm ("inclusion"). In both of these cases, it is possible to make the assertion over groups of DNS names.

The third function describes a portion of the tree that would be covered by targets containing a wildcard, but where the policy is the opposite of that expressed with the wildcard. This is expressed simply by including the relevant specific exception. For example, all the subdomains under example.com could be indicated in a target "*.example.com". To express a different policy for exception.example.com than for the rest of the names under example.com requires two SOPA RRs, one with the target "*.example.com" and the other with the target "exception.example.com". The most-specific match to a target always wins.

Is is important to note that the default setting is "exclusion". A domain name does not lie in any other name’s policy realm unless there is an explicit statement by appropriate SOPA resource record(s) to the contrary. If a candidate name does not appear in the target of any SOPA record for some owner name, then that candidate target does not lie in the same policy realm as that owner name.

It is acceptable for there to be more than one SOPA resource record per owner name in a response. Each RR in the returned RRset is treated as a separate policy statement about the original queried name (QNAME). Note, however, that the QNAME might not be the owner name of the SOPA RR: if the QNAME is an alias, then the actual SOPA owner name in the DNS database will be different than the QNAME. In other words, even though a SOPA target field is not allowed to be an an alias, when resolving the SOPA RR aliases are followed; and SOPA records are accepted transitively from the canonical name back to the QNAME.

6.1. The Exclusion Relation

A SOPA record where the relation field has value 0 states that the owner name and the target name are not in the same policy realm. While this might seem useless (given the default of exclude), a SOPA record with a relation field value of 0 can be useful in combination with a long TTL field, in order to ensure long term caching of the policy.

In addition, an important function of SOPA is to enable the explicit assertion that no other name lies in the same policy realm as the owner name (or, what is equivalent, that the owner name should be treated as a public suffix). In order to achieve this, the operator of the zone may use a wildcard target together with a relation field value of 0. See Section 6.4.

In addition, an more-specific target can be used to override a more general target (i.e. with a wildcard in the target) at the same owner name. For example,

```
      example.tld   86400 IN    SOPA  0  *.example.tld

      example.tld   86400 IN    SOPA  1  www.example.tld
```

A SOPA-using client that receives a SOPA resource record with a relation value of 0 MUST treat the owner name and the target name as lying in different policy realms.

6.2. The Inclusion Relation

A SOPA record with a relation field set to 1 is an indicator that the target name lies in the same policy realm as the owner name. In order to limit the scope of security implications, the target name and the owner name SHOULD stand in some ancestor-descendant or sibling relationship to one another. A SOPA-using client that is not prepared for inclusion relationships outside the same branch of the DNS MAY ignore such relationships and treat them as though they did not exist.

The left-most label of a target may be a wildcard record, in order to indicate that all descendant or sibling names lie in the same policy realm as the owner name. See Section 6.4.

A SOPA-using client that receives a SOPA resource record where relation is set to 1 SHOULD treat the owner name and the target name as lying in the same policy realm. If a client does not, it is likely to experience unexpected failures because the client’s policy expectations are not aligned with those of the service operator.

6.3. Interpreting DNS Responses

There are three possible responses to a query for the SOPA RRTYPE at an owner name that are relevant to determining the policy realm. The first is Name Error (RCODE=3, also known as NXDOMAIN). In this case, the owner name itself does not exist, and no further processing is needed.

The second is a No Data response [RFC2308] of any type. The No Data response means that the owner name in the QNAME does not recognize any other name as part of a common policy realm. That is, a No Data response is to be interpreted as though there were a SOPA resource record with relation value 0 and a wildcard target. The TTL on the policy in this case is the negative TTL from the SOA record, in case it is available.

The final is a response with one or more SOPA resource records in the Answer section. Each SOPA resource record asserts a relationship between the owner name and the target name, according to the functions of the SOPA RRTYPE outlined above.

Any other response is no different from any other sort of response from the DNS, and is not in itself meaningful for determining the policy realm of a name (though it might be meaningful for finding the SOPA record).

6.4. Wildcards in Targets

The special character "*" in the target field is used to match any label, but not according to the wildcard label rules in section 4.3.3 of [RFC1034]. Note that, because of the way wildcards work in the DNS, is it not possible to place a restriction to the left of a wildcard; so, for instance, example.*.example.com. does not work. In a SOPA target, it is possible to place such a restriction. In such use, a wildcard label matches exactly one label: example.*.example.com. matches the target example.foo.example.com. and example.bar.example.com., but not example.foo.bar.example.com. To match the latter, it would be necessary also to include example.*.*.example.com, which is also permitted in a target. This use of the wildcard is consistent with the use in <https://publicsuffix.org/list/>.

If a SOPA target’s first label is a wildcard label, the wildcard then matches any number of labels. Therefore, a target of *.example.com. matches both onelabel.example.com. and two.labels.example.com.; the second match would not be a match in the DNS. This use of the wildcard label does not match the public suffix list, but is included for brevity of RRsets for certain presumed-common cases. This rule is subject to more-specific matching (as discussed in Section 6.1 and Section 6.2). To simplify implementation, more-specific matches cannot have internal wildcards as described above.

The reason for these differences in wildcard-character handling is because of the purpose of the wildcard character. In DNS matching, processing happens label by label proceeding down the tree, and the goal is to find a match. But in the case of SOPA, the candidate match is presumed available, because the application would not perform a SOPA look up if there were not a different target domain at hand. Therefore, strict conformance with the DNS semantics of the wildcard is not necessary. It is useful to be able to express potential matches as briefly as possible, to keep DNS response sizes small.

Multiple leading wildcard labels (e.g. *.*.example.com.) is an error. An authoritative name server SHOULD NOT serve a SOPA RR with erroneous wildcards when it is possible to suppress them, and clients receiving such a SOPA RR MUST discard the RR. If the discarded RR is the last RR in the answer section of the response, then the response is treated as a No Data response.

It is possible for the wildcard label to be the only label in the target name. In this case, the target is "every name". This makes it trivial for an owner name to assert that there are no other names in its policy realm.

Because it would be absurd for there to be more than one SOPA RR with the same target (including wildcard target) in a SOPA RRset, a server encountering more than one such target SHOULD only serve the RR for the exclusion relation, discarding others when possible. Discarding other RRs in the RRset is not possible when serving a signed RRset. A client receiving multiple wildcard targets in the RRset MUST use only the RR with relation set to 0.

As already noted, when a SOPA RR with a wildcard target appears in the same RRset as a SOPA RR with a target that would be covered by the wildcard, the specific (non-wildcard) RR expresses the policy for that specific owner name/target pair. This way, exceptions to a generic policy can be expressed.

6.5. TTLs and SOPA RRs

The TTL field in the DNS is used to indicate the period (in seconds) during which an RRset may be cached after first encountering it (see [RFC1034]). As is noted in Section 4, however, SOPA RRs could be used to build something like the public suffix list, and that list would later be used by clients that might not themselves have access to SOPA DNS RRsets. In order to support that use as reliably as possible, a SOPA RR MAY continue to be used even after the TTL on the RRset has passed, until the next time that a SOPA RRset from the DNS for the owner name (or a No Data response) is available. It is preferable to fetch the more-current data in the DNS, and therefore if such DNS responses are available, a SOPA-aware client SHOULD use them. Note that the extension of the TTL when DNS records are not available does not extend to the use of the negative TTL field from No Data responses.

7. What Can be Done With a SOPA RR

Use of a SOPA RR enables a site administrator to assert or deny relationships between names. By the same token, it permits a a consuming client to detect these assertions and denials.

The use of SOPA RRs could either replace the public suffix list or (often more likely due to some limitations -- see Section 9) simplify and automate the management of the public suffix list. A client could use the responses to SOPA queries to refine its determinations about http cookie Domain attributes. In the absence of SOPA RRs at both owner names, a client might treat a Domain attribute as though it were omitted. More generally, SOPA RRs would permit additional steps similar to steps 4 and 5 in [RFC6265].

SOPA RRs might be valuable for certificate authorities when issuing certificates, because it would allow them to check whether two names are related in the way the party requesting the certificate claims they are.

7.1. Exclusion has Priority

In order to minimize the chance of policy associations where none exist, this memo always assumes exclusion unless there is an explicit policy for inclusion. Therefore, a client processing SOPA records can stop as soon as it encounters an exclusion record: if a parent record excludes a child record, it makes no difference whether the child includes the parent in the policy realm, and conversely. By the same token, an inclusion SOPA record that specifies a target, where the target does not publish a corresponding inclusion SOPA record, is not effective.

8. An Example Case

For the purposes of discussion, it will be useful to imagine a portion of the DNS, using the domain example.tld. A diagram of the tree of this portion is in Figure 2. In the example, the domain example.tld includes several other names: www.example.tld, account.example.tld, cust1.example.tld, cust2.example.tld, test.example.tld, cust1.test.example.tld, and cust2.test.example.tld.

```
                             tld
                              |
                              |
                     ------example -----
                    /     /   |  \      \
                   /     /    |   \      \
                  /   www  account \      cust2
              test                  \
             /   \                   cust1
         cust1   cust2
```

Figure 2

In the example, the domain tld delegates the domain example.tld. There are other possible cut points in the example, and depending on whether the cuts exist there may be implications for the use of the examples. See Section 8.1, below.

The (admittedly artificial) example permits us to distinguish a number of different roles. To begin with, there are three parties involved in the operation of services:

- o OperatorV, the operator of example.tld;
- o Operator1, the operator of cust1.example.tld;
- o Operator2, the operator of cust2.example.tld.

Since there are three parties, there are likely three administrative boundaries as well; but the example contains some others. For instance, the names www.example.tld and example.tld are in this case in the same policy realm. By way of contrast, account.example.tld might be treated as completely separate, because OperatorV might wish to ensure that the accounts system is never permitted to share anything with any other name. By the same token, the names underneath test.example.tld are actually the test-instance sites for customers. So cust1.test.example.tld might be in the same policy realm as cust1.example.tld, but test.example.tld is certainly not in the same administrative realm as www.example.tld.

Finally, supposing that Operator1 and Operator2 merge their operations, it seems that it would be useful for cust1.example.tld and cust2.example.tld to lie in the same policy realm, without including everything else in example.tld.

8.1. Examples of Using the SOPA Record for Determining Boundaries

This section provides some examples of different configurations of the example tree in Section 8, above. The examples are not exhaustive, but may provide an indication of what might be done with the mechanism.

8.1.1. Declaring a Public Suffix

Perhaps the most important function of the SOPA RR is to identify public suffixes. In this example, the operator of TLD publishes a single SOPA record:

```
tld.  86400 IN SOPA 0 *.
```

8.1.2. One Delegation, Eight Administrative Realms, Wildcard Exclusions

In this scenario, the example portion of the domain name space contains all and only the following SOPA records:

```
example.tld.  86400 IN SOPA 1 www.example.tld.

www.example.tld.  86400 IN SOPA 1 example.tld.
```

Tld is the top-level domain, and has delegated example.tld. The operator of example.tld makes no delegations. There are four operators involved: the operator of tld; OperatorV; Operator1, the operator of the services at cust1.example.tld and cust1.test.example.tld; and Operator2, the operator of the services at cust2.example.tld and cust2.test.example.tld.

In this arrangement, example.tld and www.example.tld positively claim to be within the same policy realm. Every other name stands alone. A query for an SOPA record at any of those other names will result in a No Data response, which means that none of them include any other name in the same policy realm. As a result, there are eight separate policy realms in this case: tld, {example.tld and www.example.tld}, test.example.tld, cust1.test.example.tld, cust2.test.example.tld, account.example.tld, cust1.example.tld, and cust2.example.tld.

8.1.3. One Delegation, Eight Administrative Realms, Exclusion Wildcards

This example mostly works the same way as the one in Section Section 8.1.2, but there is a slight difference. In this case, in addition to the records listed in Section 8.1.2, both tld and test.example.tld publish exclusion of all names in their SOPA records:

```
tld.  86400 IN SOPA 0 *.

test.example.tld.  86400 IN SOPA 0 *.
```

The practical effect of this is largely the same as the previous example, except that these expressions of policy last (at least) 86,400 seconds instead of the length of time on the negative TTL in the relevant SOA for the zone. Many zones have short negative TTLs because of expectations that newly-added records will show up quickly. This mechanism permits such names to express their administrative isolation for predictable minimum periods of time. In addition, because clients are permitted to retain these records during periods when DNS service is not available, a client could go offline for several weeks, and return to service with the presumption that test.example.tld is still not in any policy realm with any other name.

9. Limitations of the approach and other considerations

There are four significant problems with this proposal, all of which are related to using DNS to deliver the data.

The first is that new DNS RRTYPEs are difficult to deploy. While adding a new RRTYPE is straightforward, many provisioning systems do not have the necessary support and some firewalls and other edge systems continue to filter RRTYPEs they do not know. This is yet another reason why this mechanism is likely to be initially more useful for constructing and maintaining the public suffix list than for real-time queries.

The second is that it is difficult for an application to obtain data from the DNS. The TTL on an RRset, in particular, is usually not available to an application, even if the application uses the facilities of the operating system to deliver other parts of an unknown RRTYPE.

The third, which is mostly a consequence of the above two, is that there is a significant barrier to adoption: until browsers have mostly all implemented this, operations need to proceed as though nobody has. But browsers will need to support two mechanisms for some period of time if they are to implement this mechanism at all, and they are unlikely to want to do that. This may mean that there is no reason to implement, which also means no reason to deploy. This is made worse because, to be safe, the mechanism really needs DNSSEC, and performing DNSSEC validation at end points is still an unusual thing to do. This limitation may not be as severe for use-cases that are directed higher in the network (such as using this mechanism as an automatic feed to keep the public suffix list updated, or for the use of CAs when issuing certificates). This limitation could be reduced by using SOPA records to maintain something like the current public suffix list in an automatic fashion.

Fourth, in many environments the system hosting the application has only proxied access to the Internet, and cannot query the DNS directly. It is not clear how such clients could ever possibly retrieve the SOPA record for a name.

9.1. Handling truncation

It is possible to put enough SOPA records into a zone such that the resulting response will exceed DNS or UDP protocol limits. In such cases, a UDP DNS response will arrive with the TC (truncation) bit set. A SOPA response with the TC bit must be queried again in order to retrieve a complete response, generally using TCP. This increases the cost of the query, increases the time to being able to use the answer, and may not work at all in networks where administrators mistakenly block port 53 using TCP.

10. Security Considerations

This mechanism enables publication of assertions about administrative relationships of different DNS-named systems on the Internet. If such assertions are accepted without checking that both sides agree to the assertion, it would be possible for one site to become an illegitimate source for data to be consumed in some other site. In general, assertions about another name should never be accepted without querying the other name for agreement.

Undertaking any of the inferences suggested in this draft without the use of the DNS Security Extensions exposes the user to the possibility of forged DNS responses.

11. Internationalization Considerations

There is some discussion of how to treat targets that appear to have internationalized data in Section 5.2. Otherwise, this memo raises no internationalization considerations.

12. IANA Considerations

IANA will be requested to register the SOPA RRTYPE if this proceeds.

IANA will be requested to create a SOPA relation registry if this proceeds. The initial values are to be found in the table in Section 5.1. Registration rules should require a high bar, because it’s a one-octet field. Maybe RFC required?

13. Acknowledgements

The authors thank Adam Barth, Dave Crocker, Brian Dickson, Phillip Hallam-Baker, John Klensin, Murray Kucherawy, John Levine, Gervase Markham, Patrick McManus, Henrik Nordstrom, Yngve N. Pettersen, Eric Rescorla, Thomas Roessler, Peter Saint-Andre, and Maciej Stachowiak for helpful comments.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

14.2. Informative References

[I-D.sullivan-dbound-problem-statement]
Sullivan, A., Hodges, J., and J. Levine, "DBOUND: DNS Administrative Boundaries Problem Statement", draft-sullivan-dbound-problem-statement-01 (work in progress), July 2015.

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

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <http://www.rfc-editor.org/info/rfc2181>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <http://www.rfc-editor.org/info/rfc2308>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <http://www.rfc-editor.org/info/rfc3597>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, DOI 10.17487/RFC4395, February 2006, <http://www.rfc-editor.org/info/rfc4395>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <http://www.rfc-editor.org/info/rfc5155>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <http://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <http://www.rfc-editor.org/info/rfc5892>.

[RFC5893] Alvestrand, H., Ed. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <http://www.rfc-editor.org/info/rfc5893>.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <http://www.rfc-editor.org/info/rfc5894>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <http://www.rfc-editor.org/info/rfc6365>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <http://www.rfc-editor.org/info/rfc6672>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <http://www.rfc-editor.org/info/rfc7489>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

Appendix A. Discussion Venue

This Internet-Draft is discussed in the dbound working group: dbound@ietf.org.

Appendix B. Change History

00 to 01:

* Changed the mnemonic from BOUND to AREALM

* Added ports and scheme to the RRTYPE

* Added some motivating text and suggestions about what can be done with the new RRTYPE

* Removed use of "origin" term, because it was confusing. The document filename preserves "origin" in the name in order that the tracker doesn't lose the change history, but that's just a vestige.

* Removed references to cross-document information sharing and ECMAScript. I don't understand the issues there, but Maciej Stachowiak convinced me that they're different enough that this mechanism probably won't work.

* Attempted to respond to all comments received. Thanks to the commenters; omissions and errors are mine.

01 to 02:

* Changed mnemonic again, from AREALM to SOPA. This in response to observation by John Klensin that anything using "administrative" risks confusion with the standard administrative boundary language of zone cuts.

* Add discussion of two strategies: name-only or scheme-and-port.

- Increase prominence of utility to CAs. This use emerged in last IETF meeting.

02 to 03:

- Removed discussion of scheme-and-port, which was confusing.
- Add inclusion/exclusion/exception approach in response to comment by Phill H-B.
- Change mechanism for indicating "no others" to a wildcard mechanism.
- Added better discussion of use cases

03 to 00:

- Renamed file to get rid of "origin", which caused confusion.
- Added Jeff as co-author
- Remove exception relation; instead, more than one RR is allowed.
- Added discussion of SRV records

00 to 01:

- Failed to include change control entry
- Modest rearrangement of text, little improvement

01 to 02:

- Significant rearrangement of sections
- Large removal of text (moved to problem statement document)
- Considerably more detail in specification, including more rigorous description of RRTYPE
- Altered handling of wildcard targets
- Attempt to improve overview to make it plainer what the system does
- Clarify what use cases really work

   * Reversion to permit cross-tree use, with deployment warnings that it won’t be useful

Authors’ Addresses

Andrew Sullivan
Dyn, Inc.
150 Dow St
Manchester, NH 03101
U.S.A.

Email: asullivan@dyn.com

Jeff Hodges
PayPal
2211 North First Street
San Jose, California 95131
US

Email: Jeff.Hodges@PayPal.com