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.

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