Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

C. Deccio
Verisign Labs
J. Levine
Taughannock Networks

July 6, 2015

Concepts for Domain Name Relationships
draft-deccio-dbound-name-relationships-00

Abstract

Various Internet protocols and applications require some mechanism for identifying relationships between Domain Name System (DNS) names. In this document we provide examples of protocols and applications for which knowledge of these relationships is useful, if not required. Further we discuss the various types of domain name relationships, review current needs and solutions, and identify considerations for solution sets.

Table of Contents

1. Introduction

The use of various Internet protocols and applications has introduced the desire and need for designated relationships between Domain Name System (DNS) names, beyond the lineal relationship inherent in the names themselves. While protocols, such as that used by HTTP Cookies, have traditionally used ancestral relationships to determine allowable scope of information sharing and authorization, there is an increasing need to identify relationships between arbitrary domains.

We begin by establishing terminology and concepts, after which we discuss known applications for which the identification of domain name relationships are desirable or required. We then discuss the Public Suffix List, the primary solution for domain relationships currently available. Finally, we recommend considerations for solutions in this problem space.

2. Domain Name Concepts

For consistency in language we define terms and concepts surrounding domain names.

2.1. Domain Names

A DNS domain name is represented as sequence of dot-separated labels, such as www.example.com (i.e., comprised of labels "www", "example", and "com"). This sequence corresponds to the list of the labels formed by traversing the tree representing the domain name space, from the node representing the name itself to the root (top) of the tree ([RFC1034]). In this tree context, we thus refer to domain name's parent as the domain name formed by removing the leftmost label (i.e., the domain name corresponding to the node directly above it in the tree). The parent of www.example.com is example.com.

As there are no requirements or inferences surrounding delegation (i.e., zone cut) at any point in the DNS tree, there are no assumptions in this document about administrative boundaries drawn by delegations, unless explicitly stated otherwise. That is to say that this document considers DNS names independently from their administration, as defined by the DNS.

As noted in [RFC1034], the term "domain name" is used in contexts outside the DNS. The scope of this document is limited to domain names as defined by the DNS.

2.2. Domain Name Scope

The use of domain names in various applications over time has produced a notion of scope, which we use to refer to the general ability of arbitrary entities to register children of a domain name (i.e., create child nodes in the domain name tree). In some contexts these are called "public suffixes" or "registry controlled domains" ([RFC6265]). For example, children of the top-level domain (TLD) com, are generally registrable by arbitrary entities, which puts the com domain name in the public scope. However, com's children are typically not used in the same fashion (though certainly there are exceptions), which puts them largely in the private scope.

The children of public domain names may either be in public or private scope; likewise the children of private domain names may either be in public or private scope.

While zone cuts often exist along public/private scope boundaries (e.g., between com and example.com), they are not required at these boundaries, nor are scope boundaries required at zone cuts. In this document public/private scope is considered independent of administrative boundaries defined by the DNS (i.e., zone cuts).

The most well-known delineator of public/private scope is the Public Suffix List (PSL) [PSL], which is described later in this document.

2.2.1. Public/private Boundaries

If we consider the root domain name itself to be public, then between the root domain name and any private domain name (below), there must exist at least one boundary going from some public parent to private child. The first such boundary encountered upon downward traversal from the root is the first-level public boundary. Subsequent public-to-private boundaries are referred to as lower-level public boundaries. For example, because the com domain name is considered public, if we assume that example.com is private, then the first-level public boundary is between com and example.com. If the public.example.com domain name is considered public (i.e., children domain names can be registered by arbitrary third parties) and foo.public.example.com is a private domain name, then a lower-level public boundary exists between public.example.com and foo.public.example.com.

2.3. Domain Name Relationships

In this document two types of domain name relationships are identified: ancestry and policy. An ancestral relationship exists between two domains if one domain name is an ancestor of the other.

A policy relationship exists between two domain names if their relationship is such that application policy should treat them as equivalent. For example, the two names might be administered by the operating organization, or there might business or other relationships between the two operating entities.

In the simplest case, two domain names might be policy-related for all applications or purposes. However, it is possible that two domains are related only for explicitly defined purposes.

An ancestral relationship between two names can be identified merely by comparing the names themselves to determine whether one is a substring of the other. However, there is no inherent way to determine policy relationships neither by examination of the names themselves, nor by examining the administrative boundaries (i.e., zone cuts) defined in the DNS. This is the problem being considered in this document.

3. Policy-based Domain Name Relationships

Because policy-based domain name relationships are not inherently apparent based on the names themselves or DNS protocol, mechanisms outside the DNS namespace and base protocol are necessary to advertise and detect those relationships.

In this section we enumerate the different types of ancestral and scope relationships upon which policy-based relationships can be overlaid.

3.1. Cross-Scope Policy Relationships

If scope of one domain name is public and another is private, then it can be inferred, by the definition of their respective scopes, that there exists no policy-based relationship between the two. That is, a public domain name cannot be related, for policy purposes, to a private domain name.

Note that this doesn't prohibit policy relationships between two domain names of the same scope but having (an even number) of scope boundaries in between.

3.2. Intra-Scope Policy Relationships

We now consider the existence of a policy relationship between two domains names of the same scope.

3.2.1. Public-public Policy Relationships

The connotation of a public domain name in the context of policy is that it should not be used for purposes normally associated with private domain names. For example, it would be unreasonable to expect legitimate mail to come from an email address having the exact suffix of org.au (a domain name currently identified by [PSL] as being public). This is especially true of domain names above the first-level public boundary.

Because of this connotation, one consideration for policy amongst two domain names, both public, is that no effective relationship exists because they are ineligible by definition. Other than that, there is insufficient information from only domain names and scope alone to confirm or deny a policy relationship.

3.2.2. Private-private Policy Relationships

There are two classes of potential private-private policy relationships: ancestral and cross-domain (non-ancestral). In neither case can the presence or absence of a policy relationship be confirmed using only the names and scope information.

4. Known Applications Requiring Identification of Policy-based Domain Relationships

In this section we discuss the current state of known applications requiring identification of policy-based domain name relationships.

4.1. HTTP Cookies

Domain names are used extensively in conjunction with the Hypertext Transfer Protocol (HTTP) ([RFC7230], [RFC7231]). The domain names used in Uniform Resource Identifiers (URIs) [RFC3986] are used by HTTP clients not only for resolution to an HTTP server Internet Protocol (IP) address, but also for enforcing policy.

HTTP clients maintain local state in the form of key/value pairs known as cookies ([RFC6265]). While most often cookies are initially set by HTTP servers, HTTP clients send all cookies in HTTP requests for which the domain name in the URI is within the cookies' scope. The scope of a cookie is defined using a domain name in the "domain" attribute of the cookie. When a cookie's "domain" attribute is specified as a domain name (as opposed to an IP address), the domain name in the URL is considered within scope if it is a descendant of the "domain" attribute.

RFC 2965 [RFC2965] (now obsolete) required that the value of the "domain" field carry at least one embedded dot. This was to prohibit TLDs--which were almost exclusively public--from being associated, by policy, with other domains. Cookies having public scope would enable the association of HTTP requests across different, independently operated domains, which policy association raises concerns of user privacy and security.

In the current specification ([RFC6265]), the semantic requirements were modified to match "public suffixes" because it was recognized that TLDs are not the only domain names with public scope--and that not all TLDs are public suffixes. The notion that all TLDs are inherently public has been challenged by the many and diverse domain names that have been delegated since 2013 as part of the new generic top-level domain (gTLD) program ([NewgTLDs]).

4.2. Email sender verification

An emerging sender verification called Domain-based Message Authentication, Reporting and Conformance (DMARC) [I-D.kucherawy-dmarc-base] attempts to validate the domain name of the author's address on the message's "From:" header using the DomainKeys Identified Email (DKIM) [RFC5585] and Sender Policy Framework (SPF) [RFC7208] authentication schemes. A DKIM signature and SPF check each validate a specific domain name. For DKIM it is the domain name corresponding the DKIM signature. For SPF the domain name of the message's bounce address is validated. DMARC allows approximate matching between the author's domain and the validated domain name, where one can be an ancestor or descendant of the other.

DMARC validators are supposed to ensure that the two domain names are under the same management, the specifics of which are deliberately left out of the spec.

4.3. SSL certificate requests

Secure Socket Layer (SSL) certificate authorities typically validate certificate signing requests by sending a confirmation message to one of the WHOIS contacts for the (private scope) domain name (CA/B Ballot 74 [CA/B-Ballot-74]). In cases where there are multiple levels of delegation (i.e., crossing public/private scopes), the WHOIS contact needs to be the one for the registrant of the domain, not a higher level registration.

When an SSL certificate is for a wildcard domain name, the entire range of names covered by the wildcard needs to be under the same control. Authorities do not (knowingly) issue certificates for public domain names such as *.org.au.

5. Public Suffix List

The most well-known resource currently available for identifying public domain names is the Public Suffix List (PSL) [PSL]. The PSL is explicitly referenced as an example of an up-to-date public suffix list in [RFC6265]. The PSL was developed by Mozilla Firefox developers to further address HTTP security and privacy concerns surrounding cookie scope when the "no embedded dot" rule of [RFC2965] was the upper limit.

The PSL contains a list of known public suffixes, and includes placeholder public domains designated by "wildcard" notation in the file. A wildcard implies that all children of the wildcard's parent are in fact public domain names themselves--except where otherwise noted as a wildcard exception. For example, we use the contrived entries in Table 1 to demonstrate this use of the PSL.

Entry	Meaning
example	example is public
*.example	All children of example are public
!foo.example	foo.example is private

Table 1: Contrived PSL Entries

These entries result in the scopes shown in Table 2:

Name	Scope
example	Public
foo.example	Private
baz.foo.example	Private
bar.example	Public
baz.bar.example	Private
www.baz.bar.example	Private

Domain name scope based on the PSL entries from Table 1.

Table 2: Contrived PSL Entries

The PSL effectively identifies scope, insomuch as the list is accurate. Of the 6,823 entries in the PSL at the time of this writing, all but 50 are used to designate first-level public boundaries; the remainder designate lower-level boundaries. The

primary function of the PSL, therefore, is to delineate first-level public boundaries.

Matters of policy that can be settled simply by identifying the scope of the names in question are thus addressed by the PSL. However, the question of determining whether a policy-based relationship between intra-scope names (with the possible exception of those of public scope) are unaddressed.

5.1. Known Application Usage

The PSL is used by several browsers, including Mozilla Firefox, to identify domain names as public or private. This is used for validating the domain attribute of cookies. Additionally, it provides visual and organizational convenience for readily identifying the highest intra-scope private ancestor for a given private domain name (i.e., the child of the domain name's nearest public ancestor). This is useful for organizing names and URIs by domain name, as in bookmarks, and for highlighting key parts of URIs or certificates in the address bar or other parts of the browser interface.

Existing DMARC implementations are known to use the PSL to assert policy-based relationships between SPF- or DKIM-authenticated validated domain names and domain name corresponding to the address in the "From:" header. Such a relationship is identified if two domain names are both of private scope and share an ancestral relationship.

DMARC implementations also use the PSL to identify the highest intra-scope ancestor of a (private) domain name for the purpose of looking up the DMARC DNS record. The the appropriate ancestor name is identified it is appended to the label "_dmarc" to find the appropriate information in the DNS.

SSL certificate authorities use the PSL to ensure that wildcards are not issued for domain names having public scope.

6. Solution Considerations

The problem discussed in this document is the association of domain names for policy purposes. The PSL has been the de-facto supplementary resource utilized for identifying such relationships. The shortcomings of only having domain names and their scope (e.g., via the PSL) have been treated in Section Section 5.

An alternate paradigm for addressing the problem involves a system

wherein policy-based relationships are explicitly defined on a per-domain name (pair) basis. For scalability and dynamic response this is most effectively achieved through defining these relationships in the DNS itself, e.g., through special records included in the DNS at (or near) the domain names themselves, such as the mechanism proposed in [I-D.sullivan-domain-origin-assert]. One benefit to this paradigm is that it allows the definition of policy-based relationships between arbitrary names at any locations in the DNS domain name tree, and the notion of scope becomes moot. Another benefit is that it puts the definition of those relationships in the hands of the administrators and operators of the domain names themselves, rather than a third party.

There are several challenges with the domain name-centric paradigm as well. One challenge is that it requires correct, consistent, and coordinated efforts by affected domain name operators. The number of involved parties, moving parts, and dependencies introduces more chance for error. Additionally, having the information available online (e.g., in the DNS) means that consumption by local applications is dependent on real-time Internet connectivity, which is not always possible nor desirable.

Another solution set is that which includes both a scope definition resource (e.g., the PSL) and a mechanism for explicit definition of policy-based relationships on a per-domain name basis. In this case the scope definitions are consulted first to determine whether a policy-based relationship is possible, after which (if necessary) special domain name-specific lookups are issued to further determine whether such a relationship exists. This addresses what might be the most common issues using a central, relatively simple, and established mechanism, leaving the flexibility for additional extensibility with domain name-specific relationship definitions.

We recommend that the cost and the value of the different solution paradigms be considered when developing solutions for the problem of defining policy-based relationships between domain names. As part of this, the model of domain name relationships outlined in Section Section 2.3 should be analyzed to consider which types of relationships are most in demand, and which solutions are sufficient for the circumstances in highest demand. Such will enable an appropriate and usable balance of efficiency, robustness, flexibility, and autonomy.

7. IANA Considerations

This document includes no requests for IANA.

8. Security Considerations

This document does not specify a protocol or usage and, therefore, there are no new security considerations for it. There are security considerations for major cases in which domain boundaries are used, such as HTTP Cookies and DMARC, both discussed here. See the Security Considerations of RFC 6265 [RFC6265] and [I-D.kucherawy-dmarc-base].

9. Informative References

[CA/B-Ballot-74]
Certificate Authority(CA)/Browser Forum, "Ballot 74", 2015, <https://cabforum.org/2012/05/31/ballot-74-updates-to-domain-and-ip-validation-high-risk-requests-and-data-source-in-the-baseline-requirements/>.

[I-D.kucherawy-dmarc-base]
Kucherawy, M. and E. Zwicky, "Domain-based Message Authentication, Reporting and Conformance (DMARC)", draft-kucherawy-dmarc-base-13 (work in progress), February 2015.

[I-D.sullivan-domain-origin-assert]
Sullivan, A., "Asserting DNS Administrative Boundaries Within DNS Zones", draft-sullivan-domain-origin-assert-02 (work in progress), October 2012.

[NewgTLDs]
ICANN, "New Generic Top-Level Domains", 2015, <http://newgtlds.icann.org/>.

[PSL] Mozilla Foundation, "Public Suffix List", 2015, <https://publicsuffix.org/>.

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

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5585] Hansen, T., Crocker, D., and P. Hallam-Baker, "DomainKeys Identified Mail (DKIM) Service Overview", RFC 5585,

July 2009.

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

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, April 2014.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

Authors' Addresses

Casey Deccio
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
USA

Phone: +1 703-948-3200
Email: cdeccio@verisign.com

John Levine
Taughannock Networks
PO Box 727
Trumansburg, NY 14886

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