Defining and Signaling Relationships Between Domains

Casey Deccio John Levine

Abstract

Various Internet protocols and applications require some mechanism for determining whether two Domain Name System (DNS) names are related. In this document we formalize the types of domain name relationships, identify protocols and applications requiring such relationships, review current solutions, and describe the problems that need to be addressed.

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 define relationships between arbitrary domains.

We begin by introducing terminology, after which we discuss known and conceived applications for which domain relationships are desirable or required. We then discuss the Public Suffix List, the primary solution for domain relationships currently available. Finally, we define the problems that need addressing in the context of prior sections.

1.1 Terminology

For consistency in language we define terms surrounding domain names.

1.1.1 Domains and Domain Names

A *domain name* is a sequence of dot-separated alphanumeric words, such as `www.example.com`. A domain names *parent* is the domain name formed by removing the leftmost label. The parent of `www.example.com` is `example.com`. The *domain* of a domain name consists of the name itself and all its descendants (children, grandchildren, and so on). The names `x.y.z.example.com` and `example.com` are both in the `example.com` domain, but of the two, only `x.y.z.example.com` is in the `z.example.com` domain. The name `example.org` is in neither domain. The `root` domain name (".") is the ancestor of all domain names, and its children are referred to as *top-level domains* (TLDs). In a structural sense, the root is the top of a tree, which recursively branches to each child, and a domain is the complete subtree rooted at the name of that domain.

1.1.2 Public/private Boundaries

We refer to a domain name as having a *public* scope if the entity that operates it allows the creation of children names by other entities; otherwise, its scope is considered *private*. Domain names having public and private scope may simply be referred to as public and private, respectively. For example, many TLDs, including *com*, are public. Both public and private domains may have either public and private children, resulting in public domains within private domains and vice-versa.

1.1.3 Related Domains

We say that two domains are *related* if they are mutually and positively associated with one another. For example, they might by the same entity or that there are formal business or other relationships between the two operating entities that require a positive association. In the most lenient case, two domains, if related, are related for all purposes. However, it is possible that two domains are related for explicitly defined purposes.

1.1.4 Domain Relationships

Relationships between domains are natural, expected, and already exist. Many relationships are hierarchical, i.e., following the ancestral structure of the DNS tree, and many are not. Two related domains need not be in the same scope.

Intra-domain Relationships The most natural of relationships between domains is when one is within the other, such as a parent/child relationship. There are two major subclasses of intra-domain relationships: those that cross domain scope, and those that are within the same scope. In the former case, one or more boundaries exist between the domains. For example, assuming `example.com` is a private domain name under `com` (which is public), there is a change in scope from public to private between com and `example.com` and thus a scope boundary between the two.

Between two related domains within the same scope, there may be organizational boundaries. For example, if `foo.example.com`, with private scope, is operated by a business office of its parent organization, which operates `example.com`, there might be policy or political constraints for which an organizational boundary exists between the two.

Cross-domain Relationships Relationships between non-overlapping domains is also common. For example, `example.com` and `example.org` might be operated by the same organization, but that organization uses them interchangeably.

2 Known Applications for Domain Relationships

2.1 HTTP Cookies

The DNS is used extensively in conjunction with the Hypertext Transfer Protocol (HTTP). In addition to domain name resolution, the DNS is used by HTTP clients for enforcing policy.

HTTP clients maintain local state in the form of key/value pairs known as *cookies*. While most often cookies are initially set by HTTP servers, HTTP clients send all cookies in HTTP requests for which the domain name of the Uniform Resource Locator (URL) is within the 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 Internet Protocol address), the domain name in the URL is considered within scope if it is a descendent of the "domain" attribute.

By policy, HTTP clients are not allowed to respect a cookie domain at the TLD level or higher. Such domains have inherent public scope, and cookies having such scope would enable the association of HTTP requests across different, independently operated domains. This association raisese concerns of user privacy and security.

The TLD restriction alone is insufficient to address the privacy and security concerns with cookies. For example, many country-code TLDs (ccTLDs) designate public second-level domains (e.g., `com.ac`). Additionally, relationships are only defined within the DNS hierarchy; cross-domain relationships are not supported.

2.2 Email sender verification

An emerging sender verification called Domain-based Message Authentication, Reporting and Conformance (DMARC)[1] attempts to validate the domain name of the author's address on the message's "From:" header using the DomainKeys Identified Email (DKIM) and Sender Policy Framework (SPF) 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, where one can be an ancestor or descendant of the other.

[1]See http://www.dmarc.org/.

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

Table 1: Contrived PSL entries.

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

2.3 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[2]. 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, the entire range of names covered by the wildcard needs to be under the same control. Authorities do not (knowingly) issue certificates for public domains such as `*.com.ac`.

3 Public Suffix List

To further address HTTP security and privacy concerns surrounding cookie scope and use, a list of publix suffixes was developed by Mozilla Firefox developers. This list, known as the Public Suffix List (PSL) [3], is maintained with the Mozilla Firefox source code and includes a more extended list of known public suffixes, so cookie policies originally applied only to TLDs can be applied generally to identified public suffixes.

The PSL includes placeholder public domains designated by “wildcard” notation in the file, where a wildcard implies that all children of the wildcards 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. These entries result in the scopes shown in Table 2:

The PSL effectively identifies scope. Of the 6,823 entries in the PSL at the time of this writing, all but 50 are used to designate unbroken public scope from TLDs to the entry itself. The remainder designate public scope below a private

[2]See the Certificate Authority (CA)/Browser Forum’s Ballot 74 for a more detailed definition of “Domain Authorization Document” at https://cabforum.org/2012/05/31/ballot-74-updates-to-domain-and-ip-validation-high-risk-requests-and-data-source-in-the-baseline-requirements/

[3]See http://publicsuffix.org/.

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

Table 2: Domain name scope based on the PSL entries from Table 1.

domain. The primary function of the PSL, therefore, is to delineate the highest boundary between public and private domain space. Secondarily it identifies public/private boundaries at arbitrary levels in the DNS namespace.

The issue of determining relationships between domains that are lineally connected (i.e., one is an ancestor of another) and within the same public/private scope is not addressed by the PSL, nor is the issue of cross-domain relationships.

3.1 Usage

As alluded to previously, the PSL is used by several browsers, including Mozilla Firefox, to identify domains 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. This is useful for organizing names and URLs by domain name, as in bookmarks, and for highlighting key parts of URLs or certificates in the address bar or other parts of the browser interface.

Existing DMARC implementations are known to use the PSL to assert positive association of SPF- or DKIM-authenticated validated domain names with the domain name corrsponding to the address in the "From:" header. Inferred relationships are simply derived from intra-domain relationships for which scope boundaries are not crossed.

DMARC implementations also use the public suffix 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 public domain names.

4 Problem Statement

The PSL is the primary resource for determining public/private boundaries. Public domains in unbroken public scope, as well as those under islands of private scope, can both be designated in the PSL. However, that is the extent of its capabilities.

We propose the following with regard to domain relationships:

- Evaluate the effectiveness of the current PSL at performing the functions within its scope.
- Evaluate the effectiveness of the maintenance and distribution of the PSL.
- Determine demand for domain relationship identification other than that provided by the PSL.
- Identify solutions to replace, extend, improve, or complement the functionality, maintenance, or distribution of the PSL according to perceived demands.

4.1 Considerations

Considerations on evaluation of existing solutions and development of new solutions include the following.

Online vs. offline lookups. Online publication and detection of domain relationships (e.g., as entries in the DNS) has the advantage of rapid dissemination of changes. Fresh information can be retrieved as soon as it has been published and prior versions have expired. However, online publication can make offline lookups difficult or impossible. For a resource of capped scope, such as the current PSL, embedding and periodic synchronization of the list for offline use is feasible, e.g., through incremental or full transfer. Online schemes for detecting other relationships might not have such clear delineators, making offline use less practical. Additionally, online detection can be chatty and add latency, which might be unattractive where stealth and/or efficiency are desired.

Centralized vs. distributed maintenance and distribution. Centralized maintenance of a lookup service can be perceived as a bottleneck, particularly for the general topic of domain name relationships. If maintained in a distributed system, such as the DNS, the burden and responsibility of updating is decentralized and falls under the entity which controls the component of the service to which the lookup is directed, allowing it to scale. However, if the scope is fixed, as it is with the current PSL, scalability might not be a concern. In such cases, good practice, administered centrally, can potentially outweigh the benefits of de-centralized service. For example, version control and transparency, association of changes to a software release, and general system consistency are all exemplified by the current PSL.

Scope and scalability associated with new generic TLDs While traditionally TLDs have been implicitly considered public scope, the Internet Corporation for Assigned Names and Numbers' (ICANN's) new generic TLD (gTLD) program [4] has introduced a new class of TLDs, whose scope may differ from the norm. Depending on how delegations are handled in each, this addition might

[4]See http://newgtlds.icann.org/en/

raise new scalability questions with regard to domain name scope boundaries (e.g., that provided by the PSL) or general domain relationship identification.