The TLD Subdomain Structure Protocol and its use for Cookie domain validation
draft-pettersen-subtld-structure-02

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 10, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document defines a protocol and specification format that can be used by a client to discover how a Top Level Domain (TLD) is organized in terms of what subdomains are used to place closely related but independent domains, e.g. commercial domains in country code TLDs (ccTLD) like .uk are placed in the .co.uk subTLD domain.
This information is then used to limit which domains an Internet service can set cookies for, strengthening the rules already defined by the cookie specifications.

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].

1. Introduction

The Domain Name System [RFC1034] used to name Internet hosts allows a wide range of hierarchical names to be used to indicate what a given host is, some implemented by the owners of a domain, such as creating subdomains for certain tasks or functions, others by the Top Level Domain registry owner to indicate what kind of service the domain is, e.g. commercial, educational, government or geographic location, e.g. 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 [RFC2965] [NETSC]. Most 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, and it is often difficult to determine the source of the information being shared. To some extent [RFC2965] addresses some of these concerns for cookies, in that clients that supports [RFC2965] style cookies sends 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] is not widely deployed in clients, or on servers. The recipient(s) can make inappropriate information sharing more difficult by requiring the information to contain data identifying the source and assuring the integrity of the data, e.g. by use of cryptographic technologies. These techniques tend, however, 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.
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 webhotels, 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 a TLS Subdomain Structure Protocol that can be used to discover the actual structure of a Top Level Domain e.g. that the TLD have several subTLDs co.tld, ac.tld, org.tld, then it will show how this information can be used to determine when information sharing through cookies is not desirable.

2. The TLD Subdomain Structure Protocol

The TLD Subdomain Structure Protocol is an HTTP service, managed by the TLD owner, and located at a well known 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. Procedure for use:

- The client should retrieve the domain list for the Top Level Domain "tld" from https://www.subdomains.tld/tld/domainlist. [The actual location must be decided by IANA, this section contains the author’s suggestion. Due to security considerations it should be considered whether or not an https URL, or at least a signed file should be used]

- If the client is not able to retrieve a list from the "subdomains" server it MAY attempt to retrieve a list from a vendor specified location. Alternatively, the vendor MAY mirror the list from the "subdomains" server(s), and only retrieve the lists from the vendor specified location.

- The Content-Type of the returned list MUST be application/subdomain-structure.

- The retrieved specification SHOULD be cached 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 methods, depending on the profile.
2.1. 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 subTLD or to hide a subTLD’s classification. Beside 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 which it may not already support, which may be problematic for clients having limited resources.

2. Using 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.

2.2. Domainlist format

The domain list file can contain a list of subdomains that are considered top level 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 implicit from the request URI. The domain names listed MUST be encoded in punycode, according to [RFC3490].

2.2.1. Domainlist 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:integer }?,
    attribute name { xsd:NCName }?,
    (domain | registry)*
  }
registry =
  element registry {
    attribute levels { xsd:integer }?,
    attribute name { xsd:NCName },
    (domain | registry)*
  }
domain =
  element domain {
    attribute name { xsd:NCName }
  }

The domainlist file contains a single <tld> block, which may contain multiple registry and domain blocks, and a registry block may also contain multiple registry and domain blocks.

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

All names MUST be punycode encoded to make it possible for clients not supporting IDNA to use the document.

The tld and registry blocks MAY contain an attribute, levels, specifying how many levels below the current domain are registry-like. The default is none, meaning that the default inside the current domain level is that labels are ordinary domains and not registry-like. If the levels attribute is 1 (one) it means that by default all next-level labels within the registry/tld are registry like and not normal domains.

Implementations MUST ignore attributes and syntax they do not recognize.

2.2.2. Domainlist interpretation

For each new registry or domain block within the tld or registry the effective domain name the block applies to is the name of the block prepended to the ".name" 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
<?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="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 registry-like domain, although parliament.tld is not a registry-like 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.

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

3. A TLD Subdomain Structure Protocol profile for Cookies

HTTP State management cookies is one area 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 may also be used to track the users across services in an undesirable fashion. Neither the original Netscape cookie specification [NETSC] nor [RFC2965] are adequate in many cases.

The [NETSC] rules require only that the target domain must have one internal dot (e.g. example.com) if the TLD belong to a list of generic TLDs (gTLD), while for all 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.

[RFC2965] set the requirement that cookies can only be set for the server’s parent domain.

Unfortunately, both policies still leave open the possibility of setting cookies for a subTLD by setting the cookie from a host name example.subtld.tld to the domain subtld.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 subTLD structure like the one used by the .uk ccTLD (e.g. .co.uk), or a combination of both. But 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 servicenames from subTLDs by using DNS has been defined in [DNSCOOKIE]. However this method is not immune to TLD registries that uses subTLDs as directories, or to services that does not define an IP address for the domainname. Using the TLD Subdomain Structure Protocol to retrieve a list of all subTLDs in a given TLD will solve both those problems.

3.1. Procedure for using the TLD Subdomain Structure Protocol 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 domain structure of the TLD is not known already, or the structure information has expired, the client should retrieve or validate the structure specification from the server hosting the specification, according to section 2. If retrieval is unsuccessful, and no copy of the specification is known, the client MAY use alternative methods to decide the domain’s status, e.g. those described in [DNSCOOKIE], or other heuristics. Evaluate the specification as specified in section 2. If the target domain is part of the subTLD structure the cookie MUST be discarded.

2. If the target domain is not a subTLD, the cookie is accepted.

3.2. Unverifiable transactions

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, that can be used to select advertisements that are of interest to the user.
[RFC2965] specified that if the name of the host of an included resource does not domain match the domain reach (defined as the parent domain of the host) of the URL of the document the user started loading, loading the resource is considered an unverifiable transaction, and in such third party transactions cookies should not be sent or accepted. The latter point is not widely implemented, except when selected by especially interested users. This means that server1.example.com and server2.example.com can share cookies, and either can be referenced automatically (e.g. by including an image) by the other without being considered an unverifiable transaction, while requests to server3.example2.com would be considered an unverifiable transaction. However, like the normal domain matching rule for cookies, this rule opens up some holes. If the host example.co.uk requests a resource from server4.example3.co.uk, the request to example3.co.uk server would not be considered an unverifiable transaction because example.co.uk’s reach is co.uk, which domain matches server4.example3.co.uk, a conclusion which is obviously, to a human with some knowledge of the .uk domain structure, incorrect.

To avoid such misclassifications clients SHOULD apply the procedure specified in 3.1 for the reach domain used to decide if a request is an unverifiable, and if the reach domain is a subTLD, the reach of the original host must be changed to become the same as the name of the host itself, and requests that do not domain match the original host’s name must be considered unverifiable transactions. That is, the reach for example.co.uk becomes example.co.uk, not co.uk, and example3.co.uk will therefore not domain match the resulting reach.

4. Examples

The following examples demonstrate how the TLD Subdomain Structure Protocol can be used to decide cookie domain permissions.

4.1. Example 1

```xml
< ?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 subTLDs, except "example.tld" for which cookies are allowed. Cookies are also implicitly allowed for any y.x.tld domains.
4.2. Example 2

```xml
<?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 registry-like domains for which cookies are not allowed, for any other domains cookies are allowed.

4.3. Example 3

```xml
<?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.

5. IANA Considerations

This specification requires that the domain list is retrievable from a well-known location. This means that a hostname or group of hostnames must be assigned to serve the domain list. Suggestions for where to located the service are described in section 5.1 The specification also requires that responses are served with a specific media type. Section 5.2 provides the registration of this media type.

5.1. Location of the TLD Subdomain Structure specification

The location of the domain list must be located at a location that can easily be deduced by the client from the name of the TLD. Several possibilities exist:

1. A reserved domain name in the TLD’s name space e.g. [link]

https://www.subdomains.tld/domainlist or
https://subdomains.nictld.tld/domainlist .
2. A common repository, e.g.
   https://subdomains.example.org/tld/domainlist, managed by the
   IANA or another Internet governance body

The benefit of the first alternative is that the data are not located
at a single repository which makes it more difficult to shut down the
system completely. On the other hand the TLD registries may find the
overhead of maintaining such a service burdensome, and therefore
avoid implementing it, or let the service lapse. The second
alternative creates a common repository, which may increase adoption.
On the other hand, a single location makes it more susceptible to
denial of service attacks.

5.2. 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 6

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@opera.com

Intended usage: common

Restrictions on usage: none
6. Security Considerations

Retrieval of the specifications are 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 3 of this document describe using the domain list defined in section 2 as a method of increasing security. The effectiveness of the domain list for this purpose, and the resulting security for the client depend both on the integrity of the list, and its correctness. The integrity of the list depends on how securely it is stored at the server, and how securely it is transmitted. This specification recommends downloading the domain list using HTTP over TLS, 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 approach would be to digitally sign the files.

The correctness of the list depends on how well the TLD registry defined it. A list that does not include some subTLDs may expose the client to potential privacy and security problems, but not any worse than the situation would be without this protocol and profile, while a subdomain incorrectly classified as a subTLD 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 registry, and by interested thirdparties.

7. Acknowledgements

Anne van Kesteren assisted with defining the XML format in Section 2.2.1.

8. References
8.1. Normative References


8.2. References


Appendix A. Open issues

- Download location URI for the original domain lists
- Should Digital signatures be used on the files, instead of using TLS?
Author’s Address

Yngve N. Pettersen
Opera Software ASA
Waldemar Thranes gate 98
N-0175 OSLO,
Norway

Email: yngve@opera.com