Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2017

P. Hoffman
ICANN
J. Hildebrand

October 15, 2016

DNS Queries over HTTPS
draft-hoffman-dns-over-http-01

Abstract

This document describes how to make DNS queries and get DNS responses over HTTPS. The main driver for this document is to allow clients who want to send DNS queries over HTTP transport to be able to do in a secure and interoperable fashion, regardless of the format of the responses.

Comments on this draft can be sent to the dnsoverhttp mailing list at https://www.ietf.org/mailman/listinfo/dnsoverhttp .

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on April 18, 2017.

Table of Contents

1. Introduction

Over time, there have been many proposals for using HTTP and HTTPS as a substrate for DNS queries and responses. To date, none of those proposals have made it beyond early discussion, partially due to disagreement about what is the "best" method to do so. In particular, there has been disagreement about what the best format for the responses should be. Also, some early proposals have not followed best practices for using HTTP.

This document defines a specific protocol for sending DNS [RFC1035] queries and getting DNS responses over HTTP [RFC7230] that is running over TLS [RFC5246]. Although there might be a desire to run this protocol over an insecure transport such as bare HTTP, this document only defines the protocol as HTTP over TLS.

This design focuses on DNS stub-to-resolver communication, but DNS resolver-to-authoritative communication should work just as well.

A server that supports this protocol is called a "DNS API server" to differentiate it from a "DNS server" (one that uses the regular DNS

protocol). Similarly, a client supports this protocol is called a "DNS API client".

1.1. Use Cases

Earlier proposals for DNS over HTTP have had many different use cases. The primary use case is an application that wants to avoid network path involvement with DNS. The protocol can be implemented in the application such as a browser if the location of the DNS API server can be configured, hard-coded, or discoverable such as through DHCP.

Another use case is an operating system that wants to help applications when the OS detects broken DNS in its operations. The OS can still respond to calls such as getaddrinfo() and gethostbyname() by using this protocol without the applications needing to do anything.

A more recent use case is a small ("IoT") device that already runs COAP [RFC7252] and has a JSON [RFC7159] or CBOR [RFC7049] parser and wants to make DNS queries beyond what are supported by the device's operating system.

An eventual use case might be editing of DNS zones by end users, such as described in [I-D.hildebrand-deth]. Such editing could easily be done using existing HTTP semantics.

As HTTP/2 [RFC7540] and QUIC [I-D.hamilton-early-deployment-quic] become more widely deployed, this protocol might become more important because an HTTP/2 or QUIC server might push DNS responses to a client that the HTTP/2 server expects the client to look up. This will be covered in Section 5. TODO: this discussion of H2 push needs to be expanded by people with this use case.

These use cases assume that the server is a resolver, but this protocol can certainly be used in use cases where the server is an authoritative server. Such use cases may be added to this document, or may be documented later.

1.2. Protocol Requirements

The protocol described here bases its design on the following protocol requirements:

- The protocol must use HTTP semantics the way that they are commonly used in other protocols; there is nothing special about the DNS use case.

- The protocol must run over secure transport.
- The query format must be able to be flexible enough to express every normal DNS query.
- The response must be able to be in different formats that can be described by different documents.
- Both the query format and the response formats must be extensible. In specific, a query must be able to contain one or more EDNS extensions, including those not yet defined. Further, it must be easy to define different response formats and to extend already-defined formats.

Non-requirements:

- Supporting network-specific DNS64 [RFC6147]
- Supporting other network-specific inferences from plaintext DNS queries

1.3. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Template

The URI template (see [RFC6570]) for DNS API queries is:

{PREFIX}{QNAME}/{QTYPE}

[I-D.nottingham-json-home] will be evaluated in the future to see if the added complexity of including it in the discovery process would make for an interesting increase in deployment flexibility.

The following variables are used to expand the URI template.

2.1. PREFIX

"PREFIX" will be a URI fragment, such as "https://example.net/api/". The URI protocol MUST be "https:" or "coaps:". The prefix MUST NOT contain the character "?", and MUST be suitable for processing using the above template. Often, this will mean that the template should end with a "/" (U+002F: SOLIDUS).

The DNS API client discovers the PREFIX for the DNS API through the same mechanism as for a DNS resolver: DHCPv4, DHCPv6, IPv6 RA, and configuration. Specific PREFIX discovery mechanisms will be defined later, but imagine a new DHCP or RA option that gives a PREFIX.

If no PREFIX is configured as above, the client MAY query a DNS resolver for which they have an IP address. The query is

```
https://<IPADDRESS>/.well_known/TBD1
```

If the DNS server knows about API support, the returned URI will be the PREFIX.

The response to this discovery might be multiple PREFIXes. In that case, some of the URI types might not be supported by the resolver; this is fine as long as at least one type is. For example, if a discovery query returns both https: and coaps: URI templates, but the DNS API client can only generate https: queries, the other URI templates are ignored.

TODO: there are several concerns to be worked out with respect to the PREFIX, including how to bootstrap PREFIXes that contain domain names, and how to trust TLS connections to PREFIXes that contain only IP addresses in a deployable way. Some ideas might come from RFC 7858.

Note: The discovery response may give hints that the DNS API server requires a form of HTTP authorization. The configuration of that authorization is out of scope for the DNS API protocol. TODO: Need to think about HTTP authorization mechanisms. This would allow user tracking, but could also free resolvers from having to use IP address ranges for filtering. Several bad ideas are likely here, so let's think about it early.

2.2. QNAME

The QNAME is adopted from [RFC1035].

TODO: how to encode non-ASCII domain names. For IDNs, there are several options:

- Punycode all labels before sending
- Send the UTF8-encoded version after normalization, but before Punycoding
- Send generic UTF8-encoded labels, and make the server do normalization

TODO: domain names (not host names) that have values that might cause problems in the URI format, such as values that are control characters in ASCII, and values greater than 127.

2.3. QTYPE

The numerical QTYPE is adopted from [RFC1035].

TODO: there are some people that want to use the string forms, such as "AAAA", perhaps in addition to the numerical form. To be discussed.

2.4. Additional Parameters

The following are the names and descriptions of parameters for DNS API queries. All parameters are optional. DNS header values and extensions that are not appropriate for queries are not defined. Each parameter in the list below will need to include a use case in a later version of this document; this list is a first approximation of what may be needed.

Each of these parameters may be used in a query component of the URI sent to the API, to modify the request.

qc: QCLASS from [RFC1035] - if omitted, server assumes 1 (IN). Might be needed to support legacy DNS classes, or to access interesting new DNS capabilities.

id: ID from [RFC1035] - if omitted, there is no default value. Could be used to track responses.

opcode: Opcode from [RFC1035] - if omitted, server assumes 0 (standard query)

rd: RD from [RFC1035] - if omitted, server assumes 1 (recursion desired)

cd: CD from [RFC4035] - if omitted, server assumes 1 (DNSSec checking by the resolver disabled)

do: DO from [RFC4035] - if omitted, server assumes 1 (include RRSIG RDATA in the response)

The following are EDNS0 [RFC6891] extensions. If an extension is omitted, the server assumes that the extension was not given in the request.

nsid: Request the server's NSID, based on [RFC5001]. Set to "1" to enable.

dau: Specify the list of signing algorithms understood, based on [RFC6975]. The value is a list of integers separated by commas (with no spaces).

dhu: Specify the list of hash algorithms understood, based on [RFC6975]. The value is a list of integers separated by commas (with no spaces).

n3u: Specify the list of NSEC3 hash algorithms understood, based on [RFC6975]. The value is a list of integers separated by commas (with no spaces).

ecs: Specify the client subnet, based on [RFC7871]. The value is the bytes of the ECS option, starting with byte 4, encoded in lowercase hexadecimal.

pad: Optional padding, used for the same purposes as described in [RFC7830]. This can be used to normalize the length of queries.

See Section 6 for a registry for additional names for queries.

3. Queries

To send a DNS query, the DNS API client prepares an HTTP/CoAP GET request using the template (see Section 2). If any additional parameters (see Section 2.4) are desired, they are appended to the template as if they are HTML form data. (See Sections 4.10.22.3 and 4.10.22.4 of [HTMLSPEC] for the full specification of form data.) Typically, this is done using a "?" (U+003F: QUESTION MARK), then each parameter specified as a name, "=" (U+003D: EQUALS SIGN), and value, each name=value pair separated by a "&" (U+0026: AMPERSAND). Each value should be percent-encoded as needed. The client MUST ensure that the resulting URI is valid.

The HTTP-related requirements include:

- The HTTP GET request MUST have no body.
- The HTTP GET request SHOULD be sent with an HTTP "Accept:" header to say what type of content can be returned; of course, a server can return whatever type of content it wants. If the request does not have an HTTP "Accept:" header, the DNS API server might return a content type that the DNS API client does not understand.

- The HTTP GET request SHOULD use If-None-Match headers if earlier responses to the same query used HTTP ETag headers as described in [RFC7232].

For example, assume that the server's PREFIX is:

```
https://dnsserver.example.net/api/v1/
```

A query for the A records for "www.example.com" with recursion turned off would be:

```
https://dnsserver.example.net/api/v1/www.example.com/1?rd=0
```

The HTTP request might look like:

```
GET api/v1/www.example.com/1?rd=0 HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3
Host: dnsserver.example.net
Accept: application/dns+json
```

This document only defines the semantics of the HTTP/CoAP GET command for normal DNS queries and responses. Other verbs will be defined in the future. Other verbs will likely need different authorization semantics. For example, see [I-D.hildebrand-deth].

4. Responses

Different response formats will provide more or less information from a DNS response. For example, one response type might include the information from the DNS header bytes while another might omit it. The amount and type of information that a response format gives is solely up to the format, and not defined in this protocol.

At the time this is published, the response types are works in progress. The know response types include:

- [I-D.hoffman-dns-in-json] describes a response type in JSON
- [I-D.song-dns-wireformat-http] describes a response type in DNS wire format

In the HTTP responses, the HTTP cache headers are set to shortest DNS TTL in the response. The HTTP responses SHOULD set the HTTP ETag headers as described in [RFC7232].

TODO: Add more detail about setting the HTTP cache headers.

TODO: Add examples of creating these ETag headers.

Servers conforming to this protocol MUST implement responding with messages formatted with [I-D.hoffman-dns-in-json].

5. Use in HTTP/2

TODO: Full discussion about using this protocol in HTTP/2 for server push. This will also hopefully cover caching and DNS TTLs.

6. IANA Considerations

TODO: Create a new registry for option names for DNS queries. This will be a simple registry for new option names, probably with a designated expert.

TODO: Replace TBD1 in the body with a string from the .well_known registry. Reference [RFC5785].

7. Security Considerations

This protocol requires the use of TLS for communication. If a client does not enforce authentication of the TLS server, the communication channel will be susceptible to many security problems. See [RFC7435] for a fuller description of non-authenticated TLS.

TODO: Think about whether cross-origin resource sharing (CORS) applies to this protocol and, if so, how to specify it.

8. Acknowledgements

Early input to this document came from Mark Nottingham and and Patrick McManus.

9. References

9.1. Normative References

[HTMLSPEC] W3C, "HTML5, A vocabulary and associated APIs for HTML and XHTML", 2016, <https://www.w3.org/TR/html5/>.

[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>.

[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>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

9.2. Informative References

[I-D.hamilton-early-deployment-quic] Hamilton, R., Iyengar, J., Swett, I., and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2", draft-hamilton-early-deployment-quic-00 (work in progress), July 2016.

[I-D.hildebrand-deth] Hildebrand, J. and P. Hoffman, "DNS Editing Through HTTPS (DETH)", draft-hildebrand-deth-00 (work in progress), March 2016.

[I-D.hoffman-dns-in-json] Hoffman, P., "Representing DNS Messages in JSON", draft-hoffman-dns-in-json-09 (work in progress), October 2016.

[I-D.nottingham-json-home] Nottingham, M., "Home Documents for HTTP APIs", draft-nottingham-json-home-04 (work in progress), May 2016.

[I-D.song-dns-wireformat-http]
Song, L., Vixie, P., Kerr, S., and R. Wan, "DNS wire-format over HTTP", draft-song-dns-wireformat-http-04 (work in progress), June 2016.

[RFC5001] Austein, R., "DNS Name Server Identifier (NSID) Option", RFC 5001, DOI 10.17487/RFC5001, August 2007, <http://www.rfc-editor.org/info/rfc5001>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <http://www.rfc-editor.org/info/rfc6147>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

[RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC)", RFC 6975, DOI 10.17487/RFC6975, July 2013, <http://www.rfc-editor.org/info/rfc6975>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <http://www.rfc-editor.org/info/rfc7830>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <http://www.rfc-editor.org/info/rfc7871>.

Appendix A. Previous Work on DNS over HTTP or in Other Formats

The following is a list of earlier work that related to DNS over HTTP or representing DNS data in other formats. It is very likely incomplete, but will be expanded as this document progresses.

The list includes links to the tools.ietf.org site (because these documents are all expired) and web sites of software.

- https://tools.ietf.org/html/draft-mohan-dns-query-xml
- https://tools.ietf.org/html/draft-daley-dnsxml
- https://tools.ietf.org/html/draft-dulaunoy-dnsop-passive-dns-cof
- https://tools.ietf.org/html/draft-bortzmeyer-dns-json
- https://www.nlnetlabs.nl/projects/dnssec-trigger/

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Joe Hildebrand

Email: hildjj@cursive.net