Network Working Group P. Hoffman
Internet-Draft ICANN
Intended status: Standards Track P. McManus
Expires: November 4, 2017 Mozilla

May 03, 2017

DNS Queries over HTTPS
draft-hoffman-dns-over-https-00

Abstract

DNS queries sometimes experience problems with end to end connectivity at times and places where HTTPS flows freely.

HTTPS provides the most practical mechanism for reliable end to end communication. Its use of TLS provides integrity and confidentiality guarantees and its use of HTTP allows it to interoperate with proxies, firewalls, and authentication systems where required for transit.

This document describes how to run DNS service over HTTP using https:// URIs.

[This paragraph is to be removed when this document is published as an RFC] Comments on this draft can be sent to the DNS over HTTP 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 November 4, 2017.

Table of Contents

1. Introduction

The Internet does not always provide end to end reachability for native DNS. On-path network devices may spoof DNS responses, block DNS requests, or just redirect DNS queries to different DNS servers that give less-than-honest answers.

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 the appropriate formatting should be and partially because they did not follow HTTP best practices.

This document defines a specific protocol for sending DNS [RFC1035] queries and getting DNS responses over modern versions of HTTP [RFC7540] using https:// (and therefore TLS [RFC5246] security for integrity and confidentiality).

The described approach is more than a tunnel over HTTP. It establishes default media formatting types for requests and responses but uses normal HTTP content negotiation mechanisms for selecting alternatives that endpoints may prefer in anticipation of serving new use cases. In addition to this media type negotiation, it aligns itself with HTTP features such as caching, proxying, and compression.

The integration with HTTP provides a transport suitable for both traditional DNS clients and native web applications seeking access to the DNS.

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 that supports this protocol is called a "DNS API client".

2. 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].

3. Use Cases

There are two primary use cases for this protocol.

The primary one is to prevent on-path network devices from interfering with native DNS operations. This interference includes, but is not limited to, spoofing DNS responses, blocking DNS requests, and tracking. HTTP authentication and proxy friendliness are expected to make this protocol function in some environments where DNS directly on TLS ([RFC7858]) would not.

A secondary use case is web applications that want to access DNS information. Standardizing an HTTPS mechanism allows this to be done in a way consistent with the cross-origin resource sharing [CORS] security model of the web and also integrate the caching mechanisms of DNS with those of HTTP. These applications may be interested in

using a different media type than traditional clients.

[This paragraph is to be removed when this document is published as an RFC] Note that these use cases are different than those in a

similar protocol described at [I-D.ietf-dnsop-dns-wireformat-http]. The use case for that protocol is proxying DNS queries over HTTP instead of over DNS itself. The use cases in this document all involve query origination instead of proxying.

4. Protocol Requirements

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

- The protocol must use normal HTTP semantics.
- The query format must be able to be flexible enough to express every normal DNS query.
- The protocol must allow implementations to use HTTP's content negotiation mechanism.
- The protocol must ensure interoperable media formats through a mandatory to implement format wherein a query must be able to contain one or more EDNS extensions, including those not yet defined.
- The protocol must use a secure transport that meets the requirements for modern https://.

4.1. Non-requirements

- Supporting network-specific DNS64 [RFC6147]
- Supporting other network-specific inferences from plaintext DNS queries
- Supporting insecure HTTP
- Supporitng legacy HTTP versions

5. The HTTP Request

The URI scheme MUST be https.

The path SHOULD be "/.well-known/dns-query" but a different path can be used if the DNS API Client has prior knowledge about a DNS API service on a different path at the origin being used. (See Section 8 for the registration of this in the well-known URI registry.) Using the well-known path allows automated discovery of a DNS API Service, and also helps contextualize DNS Query requests pushed over an active HTTP/2 connection.

Some forms of the request may also include a HTTP query as defined by [RFC3986].

A DNS API Client encodes the DNS query into the HTTP request in one of two different methods:

GET: Using the on-the-wire representation of a DNS message defined in [RFC1035] as the query string portion of the URI, encoded as necessary with [RFC3986], using the GET HTTP method. This is the preferred approach because it is friendlier to HTTP caches.

POST: Including the DNS query as the message body of the HTTP request, with the request using the POST method. The body MUST use a media type selected by the DNS API Client, and that media type MUST be indicated by the request's Content-Type header.

The DNS request MAY have one or more EDNS(0) extensions [RFC6891].

The DNS API Client SHOULD include an HTTP "Accept:" request header to say what type of content can be understood in response. The client MUST be prepared to process "application/dns-udpwireformat" responses but MAY process any other type it receives.

In order to maximize cache friendliness, DNS API Clients SHOULD use the same ID (the first two bytes of the header) for every DNS request. The exact mechanism for doing so is dependent on the media type in use. HTTP semantics correlate the request and response which eliminates the need for the ID in a media type such as application/dns-udpwireformat. Using a constant value greatly increases the opportunity for successful caching.

DNS API clients can use HTTP/2 padding and compression in the same way that other HTTP/2 clients use (or don't use) them.

5.1. Example

For example, assume a DNS API server is following this specification on origin https://dnsserver.example.net/ and the well-known path. The example uses HTTP/2 formatting from [RFC7540].

A query for the IN A records for "www.example.com" with recursion turned on using the GET approach would be:

```
   :method = GET
   :scheme = https
   :authority = dnsserver.example.net
   :path = /.well-known/dns-query?%ab%cd%01%00%00%01%00%00%00%00 (no
CR)
                              %00%00%03www%07example%03com%00%00%01%00%01
   accept = application/dns-udpwireformat, application/dns-
futureJsonDns
```

The same DNS query, using the second method of HTTP encoding would be:

```
   :method = POST
   :scheme = https
   :authority = dnsserver.example.net
   :path = /.well-known/dns-query
   accept = application/dns-udpwireformat, application/dns-
futureJsonDns
   content-type = application/dns-udpwireformat
   content-length = 33

   <33 bytes represented by the following hex encoding>
   abcd 0100 0001 0000 0000 0000 0377 7777
   0765 7861 6d70 6c65 0363 6f6d 0000 0100
   01
```

6. The HTTP Response

Different response media types 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 media type 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 only known response type is "application/dns-udpwireformat", but it is likely that at least one JSON-based response format might be defined in the future.

The DNS response MAY have one or more EDNS(0) extensions, depending on the extension definition of the extensions given in the DNS request.

Native HTTP methods are used to correlate requests and responses. Responses may be returned in a different temporal order than requests were made using the protocols native multistreaming functionality.

In the HTTP responses, the HTTP cache headers SHOULD be set to expire at the same time as the shortest DNS TTL in the response. Because DNS provides only caching but not revalidation semantics, DNS over

HTTP responses should not carry revalidation response headers (such as Last-Modified: or Etag:) or return 304 responses.

A DNS API Server MUST be able to process application/dns-udpwireformat request messages.

A DNS API Server SHOULD respond with HTTP status code 415 upon receiving a media type it is unable to process.

This document does not change the definition of any HTTP response codes or otherwise proscribe their use.

6.1. Example

This is an example response for a query for the IN A records for "www.example.com" with recursion turned on. The response bears one record with an address of 93.184.216.34 and a TTL of 128 seconds.

```
:status = 200
content-type = application/dns-udpwireformat
content-length = 64
cache-control = max-age=128

<64 bytes represented by the following hex encoding>
abcd 8180 0001 0001 0000 0000 0377 7777
0765 7861 6d70 6c65 0363 6f6d 0000 0100

0103 7777 7707 6578 616d 706c 6503 636f
6d00 0001 0001 0000 0080 0004 5Db8 d822
```

7. HTTP Integration

In order to satisfy the security requirements of DNS over HTTPS, this protocol MUST use HTTP/2 [RFC7540] or its successors. HTTP/2 enforces a modern TLS profile necessary for achieving the security requirements of this protocol.

This protocol MUST be used with https scheme URI [RFC7230].

The messages in classic UDP based DNS [RFC1035] are inherently unordered and have low overhead. A competitive HTTP transport needs to support reordering, priority, parallelism, and header compression.
For this additional reason, this protocol MUST use HTTP/2 [RFC7540] or its successors.

8. IANA Considerations

This specification registers a Well-Known URI [RFC5785]:

- URI Suffix: dns-query
- Change Controller: IETF
- Specification Document(s): [this specification]

(Note for the -00 draft: a request for the media type application/dns-udpwireformat has already been submitted separately from this draft because it may be useful for other documents as well. That application is pending approval.)

9. Security Considerations

Running DNS over https:// relies on the security of the underlying HTTP connection. By requiring at least [RFC7540] levels of support for TLS this protocol expects to use current best practices for secure transport.

Session level encryption has well known weaknesses with respect to traffic analysis which might be particularly acute when dealing with DNS queries. Sections 10.6 (Compression) and 10.7 (Padding) of [RFC7540] provide some further advice on mitigations within an HTTP/2 context.

10. Acknowledgments

Joe Hildebrand contributed lots of material for a different iteration of this document. Helpful early comments were given by Ben Schwartz and Mark Nottingham.

11. References

11.1. Normative References

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <http://www.rfc-editor.org/info/rfc7858>.

11.2. Informative References

[CORS] W3C, "Cross-Origin Resource Sharing", 2014, <https://www.w3.org/TR/cors/>.

[I-D.ietf-dnsop-dns-wireformat-http] Song, L., Vixie, P., Kerr, S., and R. Wan, "DNS wire-format over HTTP", draft-ietf-dnsop-dns-wireformat-http-01 (work in progress), March 2017.

[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>.

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

The following is an incomplete list of earlier work that related to DNS over HTTP/1 or representing DNS data in other formats.

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

Patrick McManus
Mozilla

Email: pmcmanus@mozilla.com