Internet Engineering Task Force   L. Song
Internet-Draft   Beijing Internet Institute
Intended status: Experimental   P. Vixie
Expires: December 23, 2016   TISF
                                                              S. Kerr
                                                               R. Wan
                                            Beijing Internet Institute
                                                        June 21, 2016

DNS wire-format over HTTP
draft-song-dns-wireformat-http-04

Abstract

   This memo introduces a way to tunnel DNS data over HTTP. This may be
   useful in any situation where DNS is not working properly, such as
   when there is middlebox misbehavior.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 1035 [RFC1035] specifies the wire format for DNS messages. It
also specifies DNS transport on UDP and TCP on port 53, which is
still used today. However, there are other ways to access DNS
database, for example in a different data format or via alternative
DNS transport. These approaches are summarized in [draft-shane-
review-DNS-over-http].

One of alternative way of using DNS described in that document is to
transport DNS binary data inside HTTP, with the goal of improving DNS
service availability. The DNS traffic is simply sent as web traffic
using port 80/443 over HTTP. It can bypass badly behaving middle
boxes like firewalls, proxies or traffic shaping devices on path
which might interfere with normal DNS traffic [RFC5625] [DOTSE]
[SAC035].

This approach has the advantage that HTTP usually makes it through
even the worst coffee shop or hotel room firewalls, as Internet users
expect web browsing to always work. It also benefits from HTTP's
support for persistent TCP connections (see section 6.3 in
[RFC7230]). Note that 5966bis [I-D.ietf-dnsop-5966bis] specifies the
persistent feature for DNS on TCP port 53, but current DNS software
does not generally support this mode of operation. Finally, HTTPS
provides data integrity and privacy.

One alternative idea is to simply use a VPN, rather than a custom
protocol for this purpose. While this is possible, the DNS over HTTP
wire format protocol presented here works on an actual HTTP server,
so it can be hosted on a machine that also serves web pages. This

means that DNS over HTTP is slightly more "stealthy" than a VPN, in
that it can be indistinguishable from normal web traffic.

Unlike a REST DNS API using JSON [I-D.bortzmeyer-dns-json] or XML
[I-D.mohan-dns-query-xml] encoding for DNS data, in this approach
wire-format data is wrapped with a HTTP header and transmitted on
port 80 or 443. The protocol is intended to serve as a sort of DNS
VPN, and does not introduce another format of DNS data. It is also
possible as a new DNS APIs for advanced usage in application
development. For example, web developers can create arbitrary HTTP/
HTTPS queries and get DNS responses in their JavaScript apps.

This memo aims to describe how the DNS binary over HTTP concept
works. Hopefully implementations by different developers following
this memo can speak with each other.

This mechanism is designed for client stub resolver to recursive
server. DNS zone transfer, DNS updates, and anything other than
simple DNS queries are out-of-scope for this document.

2. Methodology and Configuration

As mentioned in introduction, the basic methodology is wrapping the
DNS wire-format data into an HTTP header and transmitting on port 80
or 443. However, there are two different scenarios for
implementation.

Scenario 1:

The DNS server implementation handles DNS queries and responses via
both UDP and TCP on port 53, and HTTP on port 80 or 443. It works as
follows:

1. The client creates a DNS query message.

2. The client encapsulates the DNS message in a HTTP message body
    and assigns parameters with the HTTP header.

3. The client connects to the server and issues an HTTP POST request
    method. This may re-use an existing HTTP connection.

4. The server decapsulates the HTTP packet to get the DNS query, and
    resolves the DNS query.

5. The server encapsulates the DNS response in HTTP and sends it
    back via the HTTP session.

Scenario 2:

In this scenario there is a DNS-HTTP proxy sitting between stub-resolver and the recursive server. The stub uses a client proxy and the recursive server uses a server proxy. This works like a DNS VPN and transmits wire-format DNS messages over HTTP between the proxy client and a server, as follows:

1. The stub-resolver sends a DNS query (over UDP or TCP) to the proxy client.

2. The proxy client encapsulates the DNS message in an HTTP message body and assigns parameters with the HTTP header.

3. The proxy client connects to the proxy server and issues an HTTP POST request method. This may re-use an existing HTTP connection.

4. The proxy server decapsulates the HTTP packet to get the DNS query, and sends it to a real DNS server over UDP/TCP.

5. The proxy server encapsulates the DNS response in HTTP and sends it back via the HTTP session.

6. The proxy client decapsulates the DNS message from the HTTP response and sends it back to the stub-resolver via previous DNS session (either UDP or TCP).

It is possible that these scenarios are mixed. The server may speak DNS over HTTP directly and the client use a proxy, or the other way around.

Note that the proxy client can be implemented listening to a loop-back address in the same host with stub-resolver. The proxy server can be implemented as a caching server as well. It is also possible to use the proxy server as a regular web server at the same time that is acting as a proxy server.

3. DNS-over-HTTP Message Format

DNS over HTTP is not tied to a specific version of HTTP, and should work with HTTP 1.1 [RFC7230] and HTTP/2 [RFC7540]. This section describes the details of the DNS over HTTP message format.

3.1. Request Method

A DNS message is sent over HTTP from the client to the server via a properly-formed HTTP request. This is a POST method request [section 4.3.3 in RFC 7231 [RFC7231]]. If a GET method request is sent to the

server, it optionally returns a human-readable page showing
information targeted at users.

Note that choosing POST (and not GET) as the request method for DNS
wire-format over HTTP is mainly based on two reasons. One is that
the protocol is designed using HTTP as a tunnel-like technology
carrying data from one side to another, not a web service with
RESTful framework. Another is that from the view of implementation
some servers or middleboxes may ignore an undefined entity-body if
using GET; and HTTP libraries have varying support for using GET with
a payload.

The target URI is provided explicitly by the services provider.
Derived from the target URI, the request-target in request-line
identifies the target resource upon which to apply the request. To
avoid URI conflicts and enhance interoperability, DNS wire-format
over HTTP uses a well-known URI. As defined in RFC 5785 [RFC5785],
it begins with the characters "/.well-known/" and the name "dns-
wireformat". So the request-target for DNS wire-format over HTTP
SHOULD be '/.well-known/dns-wireformat'.

A DNS transaction over HTTP has no specific requirements for the
transport protocol; developers can use any version of HTTP to
accomplish the transaction. But developers should be aware that HTTP
1.1 [RFC7230] and HTTP/2 [RFC7540] do have differences in performance
regarding multiplexing. HTTP/2 is fully multiplexed, instead of
ordered and blocking. Because there is a general desire to achieve
similar performance with DNS over UDP, the modern HTTP/2 is preferred
for DNS over HTTP implementation. Note that there should be no
problem for advanced HTTP protocol in the future deployed for DNS
over HTTP.

3.2. Response Status Code

The status code [Section 6 of [RFC7231]], only reflects the status of
the HTTP connection. If the request has succeeded, the status code
is 200 (OK).

If the request fails, the proxy server will supply an appropriate
error code, typically 4xx (client error) if the client has provided a
query that the server cannot understand for some reasons, or 5xx
(server error) if some server-side problem prevented a query from
succeeding.

To be clear, a failure on the DNS side should result in a status code
of 200 (OK) as long as the HTTP request is valid and can be answered.
The DNS error will be returned via the encapsulated DNS message.

3.3. Header Fields

   By definition header fields are key:value pairs that can be used to
   communicate data about the message, its payload, the target resource,
   or the connection (for example control data).

   The Content-Type: header field should be set to "application/dns-
   wireformat".

   We add one a new header field:

   Proxy-DNS-Transport: xyz

   Where xyz is either UDP or TCP, which is the client's indication of
   how it received the underlying DNS query, and which the server will
   use when sending the query to the far-end DNS server. This means if
   a stub DNS client asks for TCP, then that's what the far-end DNS
   server will see, and likewise for UDP.

   Exposing the transport protocol of the query allows the HTTP server
   proxy to send DNS queries to the recursive resolver that look like
   those that the DNS client is sending to the client proxy. If the
   stub resolver sent a UDP packet, then this allows the recursive
   resolver to implement the logic of truncating the packet properly,
   instead of requiring that the HTTP client proxy somehow manage that
   functionality.

   For a stub resolver that connects directly via HTTP to the HTTP
   server proxy then this flag should be set to TCP, as the entire
   response can always be delivered so truncation is never required.

   The client MUST include this option. If it is missing then it is an
   error and the server should respond with HTTP code 400 (bad request).

3.4. Message Body

   As mentioned, the message body is DNS wire-format data. It is worth
   mentioning that DNS messages sent over TCP connections is prefixed
   with a two-byte length field which gives the message length [section
   4.2.2 in RFC 1035 [RFC1035]], excluding the two-byte length field.
   This length field allows the low-level processing to assemble a
   complete message before beginning to parse it. In the context of
   HTTP, there is content-length header field [section 3.3.2 in
   [RFC7230]]in which the field-value is the same with two bytes length
   field in DNS over TCP.

   Since this two-byte length field is redundant, when the client proxy
   receives a DNS message over TCP, it MUST NOT include the length field

in the message sent to the server. The length in the content-length
header is only the size of the DNS message itself, and MUST NOT
include this two-byte length header.

[4]. Security Considerations

This protocol does not introduce any new security considerations
since it is built on the DNS and HTTP protocols.

Since this protocol transmits DNS messages, all of the security
concerns that stub resolvers and recursive resolvers have with DNS
messages apply. However, since HTTP uses TCP as the underlying
protocol, DNS reflection or amplification attacks are not possible.

Since HTTP is used, all of the security concerns of HTTP are also
present.

Servers and clients SHOULD use TLS for communication. It provides
privacy and integrity for HTTP sessions. If TLS is used, then all of
the security concerns of TLS are also present.

As specified in RFC 5246 [RFC5246], both the HTTP server and client
can be authenticated or not authenticated. Clients SHOULD
authenticate the certificate of the HTTP server they connect to. The
DNS service providers can decide whether to authenticate the client
side based on their own requirement for security and privacy. For
example, clients of an open resolver do not require authentication.

Note that the ability to perform DNS queries in this way may allow
users to bypass local DNS policy. This is problematic in any
environment where administrators need to enforce specific DNS
behavior, such as an enterprise environment. The protocol outlined
here does not introduce any new capabilities in this area, but by
creating a more standardized way of doing this it may cause
operational problems for enterprise administrators.

[5]. IANA considerations

Registration for a new Media Type: dns-wireformat

Registration for a new HTTP header field: Proxy-DNS-Transport

Registration for a new Well-Known URI: "dns-wireformat"

6. Acknowledgments

Thanks to Bob Harold, Paul Hoffman, and Julian Reschke for review.

7. References

[DOTSE]   and , "DNSSEC Tests of Consumer Broadband Routers",
            February 2008,
            <http://www.iis.se/docs/Routertester_en.pdf>.

[I-D.bortzmeyer-dns-json]
            Bortzmeyer, S., "JSON format to represent DNS data",
            draft-bortzmeyer-dns-json-01 (work in progress), February
            2013.

[I-D.ietf-dnsop-5966bis]
            Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and
            D. Wessels, "DNS Transport over TCP - Implementation
            Requirements", draft-ietf-dnsop-5966bis-04 (work in
            progress), November 2015.

[I-D.mohan-dns-query-xml]
            Parthasarathy, M. and P. Vixie, "Representing DNS messages
            using XML", draft-mohan-dns-query-xml-00 (work in
            progress), September 2011.

[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>.

[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>.

[RFC5625]   Bellis, R., "DNS Proxy Implementation Guidelines",
            BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009,
            <http://www.rfc-editor.org/info/rfc5625>.

[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>.

   [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Semantics and Content", RFC 7231,
              DOI 10.17487/RFC7231, June 2014,
              <http://www.rfc-editor.org/info/rfc7231>.

   [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>.

   [SAC035]   ICANN Security and Stability Advisory Committee, "DNSSEC
              Impact on Broadband Routers and Firewalls", 2008.

Authors' Addresses

   Linjian Song
   Beijing Internet Institute
   2508 Room, 25th Floor, Tower A, Time Fortune
   Beijing 100028
   P. R. China

   Email: songlinjian@gmail.com
   URI: http://www.biigroup.com/

   Paul Vixie
   TISF
   11400 La Honda Road
   Woodside, California 94062
   US

   Email: vixie@tisf.net
   URI: http://www.redbarn.org/

   Shane Kerr
   Beijing Internet Institute
   2/F, Building 5, No.58 Jinghai Road, BDA
   Beijing 100176
   CN

   Email: shane@biigroup.cn
   URI: http://www.biigroup.com/

Runxia Wan
Beijing Internet Institute
2508 Room, 25th Floor, Tower A, Time Fortune
Beijing 100028
P. R. China

Email: rxwan@biigroup.cn
URI: http://www.biigroup.com/