Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: June 9, 2016
December 7, 2015

S. Kerr
L. Song
R. Wan
Beijing Internet Institute

A review of implementation DNS over port 80/443
draft-shane-review-dns-over-http-00

Abstract

The default DNS transport uses UDP on port 53. There are many motivations why users or operators may prefer to avoid sending DNS traffic in this way. A common solution is to use port 80 or 443; with plain TCP, TLS-encrypted TCP, or full HTTP(S). This memo reviews the possible approaches and delivers some useful information for developers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Name servers use port 53, on both UDP and TCP [RFC1035] [RFC5966]. However, users or operators occasionally find it useful to use an alternative way to deliver DNS information, and often pick port 80 (the default HTTP port) or 443 (the default HTTPS port) for this purpose.

There are several use cases:

- Case 1: Firewalls or other middleboxes may interfere with normal DNS traffic [RFC3234] [RFC5625] [DOTSE] [SAC035].

- Case 2: Clients may want to chose a resolver other than the one locally available. For example, some ISPs and hotels perform DNS rewriting to send users to advertising pages, or may use IP addresses which cause misleading geographic location for the user [I-D.ietf-dnsop-edns-client-subnet]. Users may want DNSSEC support which is not deployed locally, and so on.

- Case 3: Users may use DNS over TLS or HTTPS to protect privacy.

- Case 4: Developers may want a higher level DNS API. Web developers may prefer different abstractions or familiar tools like JSON or XML, transmitted using HTTP or HTTPS.

This memo does not aim to develop standards or tools. The purpose is to review various implementation options as a reference for developers. However, it may be helpful for anyone hoping to develop specifications for DNS over 80/443 in the future.

Note that most of the implementations described in this memo are on port 80/443 and combined with TCP/TLS/HTTP(S). The main focus is

between stub resolvers and recursive servers, and the discussion is about the stub resolver to recursive server communication.

2. Different Implementations Approaches

2.1. DNS over TCP on port 80/443

The simplest approach is just moving the DNS traffic to port 80 or 443 from 53. This approach serves the requirement use case 1.

In this way, the whole protocol is the same as current DNS transport in TCP, except the transport port is moved to port 80 or 443. The difference between port 80 and 443 is that the traffic of port 80 is usually intercepted as HTTP traffic while the traffic of port 443 is usually considered to be encrypted, and typically ignored by middle-boxes. One example where DNS is transported through port 80/443 is one of the fallback cases of NLnetLabs' DNSSEC trigger [dnssec-trigger].

Transporting DNS through port 80/443 is an easy implementation. Developers can simply run an existing DNS server and configure the DNS software to listen on ports 80/443. The client can also apply this change without any significant changes.

One drawback of this approach is that it might mislead the client because of the port used. For example, clients might think DNS over 443 as a secure protocol because normally the session would be encrypted. In this case, however, it is not.

2.2. DNS over TLS on port 443

Another approach is DNS over TLS on port 443, which is also implemented in DNSSEC trigger. It is similar to [I-D.ietf-dprive-dns-over-tls], which uses the well-known port 853. Using port 443 to carry the traffic still serves the purpose in use case 1, as some middle boxes may block traffic on the new port.

<< Note: or we can just recommend developer to follow draft-ietf-dprive-dns-over-tls but using port 443 >>

As specified in RFC 5246 [RFC5246], both the DNS server and client can be authenticated or not authenticated. The DNS service providers can decide authenticated pattern on both server and client sides based on their own requirement for security and privacy.

TLS provide many benefits for DNS. First, it significantly reduces DNS conversation's vulnerability of being hijacked. Second, it prevents resolvers from coming amplifier of reflection attack.

Additionally, it provides privacy by encrypted the conversation between client and server.

One concern of DNS over TLS is its cost. Compared to UDP, DNS-over-TCP requires an additional round-trip-time (RTT) of latency to establish a TCP connection. Use of TLS encryption algorithms adds an additional RTT, and results in slightly higher CPU usage. It should also be considered that the DNS packet over TLS on a new port might be dropped by some middle boxes. Another concern of TLS is the deployment difficulty when authenticating the server. If servers are authenticated, certificate management is required.

2.3. DNS Wire-format over HTTP(S)

Different from DNS over TCP using port 80/443, another option is encapsulating DNS wire-format data into an HTTP body and sending it as HTTP(S) traffic. It is quiet useful in use cases 1 & 2 described in the introduction. This approach has the benefit that HTTP usually makes it through even the worst coffee shop or hotel room firewalls, as this expected by Internet users. It also benefits from HTTP's persistent TCP connection pool concept (see section 6.3 in [RFC7230]), which DNS on TCP port 53 does not have. Finally, as with DNS over TLS, HTTPS provides data integrity and privacy.

The basic methodology works as follows:

1. The client creates a DNS query message.

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

3. The client connects to the server and issues an HTTP(S) POST request method.

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

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

Note that if the original DNS query is sent by TCP, first two bits of the package is the message length and should be removed. (This is only true if some software is translating from the DNS protocol to DNS over HTTP, for example via a proxy. Native implementations will of course not need this.) There is an implementation of this methodology in the Go Programming Language (https://github.com/BII-Lab/DNSoverHTTPinGO) as well as C (https://github.com/BII-Lab/DNSoverHTTP), maintained by BII lab.

In addition to the benefits mentioned before, the HTTP header makes DNS wire-format over HTTP(S) easy to extend. Compared to creating a new option in EDNS0, using new parameters in HTTP header is far easier to deploy, since DNS messages with EDNS0 may not pass some middle boxes.

One disadvantage of packaging DNS into HTTP is its cost. Doing pack and unpack costs CPU and may result in higher response time. The DNS over HTTP messages also have a risk of being dropped by some firewalls which intercepts HTTP packets. And it should be noted that if HTTPS is used here, then all the discussion of TLS in previous section is also applicable here.

2.4. REST HTTP API

As mentioned in use case 4, one motivation of a REST HTTP API is for web developers who need to get DNS information but cannot create raw requests (for example JavaScript developers). They can work by creating HTTP requests other than real DNS queries.

In this style of implementation DNS data is exchanged in other formats than wire format, like JSON [I-D.bortzmeyer-dns-json], or XML [I-D.mohan-dns-query-xml]. There are also lots of REST DNS API developed by DNS service providers.

Most of these APIs are developed in the scope of their own system with different specification. But a typical query is a client will requesting a special formatted URI. Usually there is a HTTP(S) server listening to port 80/443, which will parse the request and create a DNS query or DNS operation command towards the real DNS. Unlike wire-format DNS over HTTP(S), once the HTTP(S) server receives the response, it create the response by putting DNS data into various structured formats like JSON, XML, YAML, or even plain text.

However, this approach may have issues, because it is not based on traditional DNS protocol. So there is no guarantee of the protocol's completeness and correctness. The support of DNSSEC might also be a problem cause the response usually do not contain RR records with the answer, making it impossible for a client to validate the reply.

3. References

[dnssec-trigger]
"Dnssec-Trigger", <https://www.nlnetlabs.nl/projects/dnssec-trigger/>.

[DOTSE] Aehlund, J. and P. Wallstroem, "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-edns-client-subnet] Contavalli, C., Gaast, W., tale, t., and W. Kumari, "Client Subnet in DNS Queries", draft-ietf-dnsop-edns-client-subnet-04 (work in progress), September 2015.

[I-D.ietf-dprive-dns-over-tls] Zi, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "DNS over TLS: Initiation and Performance Considerations", draft-ietf-dprive-dns-over-tls-01 (work in progress), October 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>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[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>.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <http://www.rfc-editor.org/info/rfc5966>.

[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>.

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

Authors' Addresses

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/

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/

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/