Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

C. Huitema
Private Octopus Inc.
A. Mankin
Salesforce
S. Dickinson
Sinodun IT

February 22, 2021

Specification of DNS over Dedicated QUIC Connections
draft-ietf-dprive-dnsoquic-02

Abstract

This document describes the use of QUIC to provide transport privacy for DNS. The encryption provided by QUIC has similar properties to that provided by TLS, while QUIC transport eliminates the head-of-line blocking issues inherent with TCP and provides more efficient error corrections than UDP. DNS over QUIC (DoQ) has privacy properties similar to DNS over TLS (DoT) specified in RFC7858, and latency characteristics similar to classic DNS over UDP.

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 https://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 August 26, 2021.

Copyright Notice

Table of Contents

1. Introduction

Domain Name System (DNS) concepts are specified in "Domain names - concepts and facilities" [RFC1034]. The transmission of DNS queries and responses over UDP and TCP is specified in "Domain names - implementation and specification" [RFC1035]. This document presents a mapping of the DNS protocol over the QUIC transport [I-D.ietf-quic-transport] [I-D.ietf-quic-tls]. DNS over QUIC is referred here as DoQ, in line with the "Terminology for DNS Transports and Location" [I-D.ietf-dnsop-terminology-ter]. The goals of the DoQ mapping are:

1. Provide the same DNS privacy protection as DNS over TLS (DoT) [RFC7858]. This includes an option for the client to authenticate the server by means of an authentication domain name as specified in "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310].

2. Provide an improved level of source address validation for DNS servers compared to classic DNS over UDP.

3. Provide a transport that is not constrained by path MTU limitations on the size of DNS responses it can send.

4. Explore the characteristics of using QUIC as a DNS transport, versus other solutions like DNS over UDP [RFC1035], DoT [RFC7858], or DNS over HTTPS (DoH) [RFC8484].

In order to achieve these goals, the focus of this document is limited to the "stub to recursive resolver" scenario also addressed by DoT [RFC7858]. That is, the protocol described here works for queries and responses between stub clients and recursive servers. The specific non-goals of this document are:

1. No attempt is made to support AXFR "DNS Zone Transfer Protocol (AXFR)" [RFC5936] or IXFR "Incremental Zone Transfer in DNS" [RFC1885], as these mechanisms are not relevant to the stub to recursive resolver scenario. (This may change in future versions

of this draft. See Appendix A for a discussion of changes required for AXFR support.)

2. No attempt is made to evade potential blocking of DNS over QUIC traffic by middleboxes.

3. No attempt to support server initiated transactions, are these are not relevant for the "stub to recursive resolver" scenario, see Section 4.5.

Users interested in zone transfers should continue using TCP based solutions and will also want to take note of work in progress to support "DNS Zone Transfer-over-TLS" [I-D.ietf-dprive-xfr-over-tls].

Specifying the transmission of an application over QUIC requires specifying how the application's messages are mapped to QUIC streams, and generally how the application will use QUIC. This is done for HTTP in "Hypertext Transfer Protocol Version 3 (HTTP/3)"[I-D.ietf-quic-http]. The purpose of this document is to define the way DNS messages can be transmitted over QUIC.

In this document, Section 4 presents the reasoning that guided the proposed design. Section 5 specifies the actual mapping of DoQ. Section 6 presents guidelines on the implementation, usage and deployment of DoQ.

2. Key Words

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 BCP 14 [RFC8174].

3. Document work via GitHub

(THIS SECTION TO BE REMOVED BEFORE PUBLICATION) The Github repository for this document is at https://github.com/huitema/dnsoquic. Proposed text and editorial changes are very much welcomed there, but any functional changes should always first be discussed on the IETF DPRIVE WG (dns-privacy) mailing list.

4. Design Considerations

This section and its subsection present the design guidelines that were used for DoQ. This section is informative in nature.

4.1. Scope is Limited to the Stub to Resolver Scenario

Usage scenarios for the DNS protocol can be broadly classified in three groups: stub to recursive resolver, recursive resolver to authoritative server, and server to server. This design focuses only on the "stub to recursive resolver" scenario following the approach taken in DoT [RFC7858] and "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310].

QUESTION: Should this document specify any aspects of configuration of discoverability differently to DoT?

No attempt is made to address the recursive to authoritative scenarios. Authoritative resolvers are discovered dynamically through NS records. It is noted that at the time of writing work is ongoing in the DPRIVE working group to attempt to address the analogous problem for DoT [I-D.ietf-dprive-phase2-requirements]. In the absence of an agreed way for authoritative to signal support for QUIC transport, recursive resolvers would have to resort to some trial and error process. At this stage of QUIC deployment, this would be mostly errors, and does not seem attractive. This could change in the future.

The DNS protocol is also used for zone transfers. In the AXFR zone transfer scenario [RFC5936], the client emits a single AXFR query, and the server responds with a series of AXFR responses. This creates a unique profile, in which a query results in several responses. Supporting that profile would complicate the mapping of DNS queries over QUIC streams. Zone transfers are not used in the stub to recursive scenario that is the focus here, and seem to be currently well served by using DNS over TCP. There is no attempt to support either AXFR or IXFR in this proposed mapping of DNS to QUIC.

4.2. Provide DNS Privacy

DoT [RFC7858] defines how to mitigate some of the issues described in "DNS Privacy Considerations" [RFC7626] by specifying how to transmit DNS messages over TLS. The "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310] specify Strict and Opportunistic Usage Profiles for DoT including how stub resolvers can authenticate recursive resolvers.

QUIC connection setup includes the negotiation of security parameters using TLS, as specified in "Using TLS to Secure QUIC" [I-D.ietf-quic-tls], enabling encryption of the QUIC transport. Transmitting DNS messages over QUIC will provide essentially the same privacy protections as DoT [RFC7858] including Strict and

Opportunistic Usage Profiles [RFC8310]. Further discussion on this is provided in Section 9.

4.3. Design for Minimum Latency

QUIC is specifically designed to reduce the delay between HTTP queries and HTTP responses. This is achieved through three main components:

1. Support for 0-RTT data during session resumption.

2. Support for advanced error recovery procedures as specified in "QUIC Loss Detection and Congestion Control" [I-D.ietf-quic-recovery].

3. Mitigation of head-of-line blocking by allowing parallel delivery of data on multiple streams.

This mapping of DNS to QUIC will take advantage of these features in three ways:

1. Optional support for sending 0-RTT data during session resumption (the security and privacy implications of this are discussed in later sections).

2. Long-lived QUIC connections over which multiple DNS transactions are performed, generating the sustained traffic required to benefit from advanced recovery features.

3. Fast resumption of QUIC connections to manage the disconnect-on-idle feature of QUIC without incurring retransmission time-outs.

4. Mapping of each DNS Query/Response transaction to a separate stream, to mitigate head-of-line blocking. This enables servers to respond to queries "out of order". It also enables clients to process responses as soon as they arrive, without having to wait for in order delivery of responses previously posted by the server.

These considerations will be reflected in the mapping of DNS traffic to QUIC streams in Section 5.2.

4.4. No Specific Middlebox Bypass Mechanism

The mapping of DNS over QUIC is defined for minimal overhead and maximum performance. This means a different traffic profile than HTTP3 over QUIC. This difference can be noted by firewalls and middleboxes. There may be environments in which HTTP3 over QUIC will

be able to pass through, but DoQ will be blocked by these middle boxes.

4.5. No Server Initiated Transactions

As stated in Section 1, this document does not specify support for server initiated transactions because these are not relevant for the "stub to recursive resolver" scenario. Note that "DNS Stateful Operations" (DSO) [RFC8490] are only applicable for DNS over TCP and DNS over TLS. DSO is not applicable to DNS over HTTP since HTTP has its own mechanism for managing sessions, and this is incompatible with the DSO; the same is true for DNS over QUIC.

5. Specifications

5.1. Connection Establishment

DoQ connections are established as described in the QUIC transport specification [I-D.ietf-quic-transport]. During connection establishment, DoQ support is indicated by selecting the ALPN token "doq" in the crypto handshake.

5.1.1. Draft Version Identification

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Only implementations of the final, published RFC can identify themselves as "doq". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier. For example, draft-ietf-dprive-dnsoquic-00 is identified using the string "doq-i00".

5.1.2. Port Selection

By default, a DNS server that supports DoQ MUST listen for and accept QUIC connections on the dedicated UDP port TBD (number to be defined in Section 10), unless it has mutual agreement with its clients to use a port other than TBD for DoQ. In order to use a port other than TBD, both clients and servers would need a configuration option in their software.

By default, a DNS client desiring to use DoQ with a particular server MUST establish a QUIC connection to UDP port TBD on the server, unless it has mutual agreement with its server to use a port other

than port TBD for DoQ. Such another port MUST NOT be port 53 or port 853. This recommendation against use of port 53 for DoQ is to avoid confusion between DoQ and the use of DNS over UDP [RFC1035]. Similarly, using port 853 would cause confusion between DoQ and DNS over DTLS [RFC8094].

5.2. Stream Mapping and Usage

The mapping of DNS traffic over QUIC streams takes advantage of the QUIC stream features detailed in Section 2 of the QUIC transport specification [I-D.ietf-quic-transport].

The stub to resolver DNS traffic follows a simple pattern in which the client sends a query, and the server provides a response. This design specifies that for each subsequent query on a QUIC connection the client MUST select the next available client-initiated bidirectional stream, in conformance with the QUIC transport specification [I-D.ietf-quic-transport].

The client MUST send the DNS query over the selected stream, and MUST indicate through the STREAM FIN mechanism that no further data will be sent on that stream.

The server MUST send the response on the same stream, and MUST indicate through the STREAM FIN mechanism that no further data will be sent on that stream.

Therefore, a single client initiated DNS transaction consumes a single stream. This means that the client's first query occurs on QUIC stream 0, the second on 4, and so on.

5.2.1. Transaction Errors

Peers normally complete transactions by sending a DNS response on the transaction's stream, including cases where the DNS response indicates a DNS error. For example, a Server Failure (SERVFAIL, [RFC1035]) SHOULD be notified to the initiator of the transaction by sending back a response with the Response Code set to SERVFAIL.

If a peer is incapable of sending a DNS response due to an internal error, it may issue a QUIC Stream Reset with error code DOQ_INTERNAL_ERROR. The corresponding transaction MUST be abandoned.

5.3. DoQ Error Codes

The following error codes are defined for use when abruptly terminating streams, aborting reading of streams, or immediately closing connections:

DOQ_NO_ERROR (0x00): No error. This is used when the connection or stream needs to be closed, but there is no error to signal.

DOQ_INTERNAL_ERROR (0x01): The DoQ implementation encountered an internal error and is incapable of pursuing the transaction or the connection

5.4. Connection Management

Section 10 of the QUIC transport specification [I-D.ietf-quic-transport] specifies that connections can be closed in three ways:

- idle timeout
- immediate close
- stateless reset

Clients and servers implementing DNS over QUIC SHOULD negotiate use of the idle timeout. Closing on idle timeout is done without any packet exchange, which minimizes protocol overhead. Per section 10.2 of the QUIC transport specification, the effective value of the idle timeout is computed as the minimum of the values advertised by the two endpoints. Practical considerations on setting the idle timeout are discussed in Section 6.6.2.

Clients SHOULD monitor the idle time incurred on their connection to the server, defined by the time spent since the last packet from the server has been received. When a client prepares to send a new DNS query to the server, it will check whether the idle time is sufficient lower than the idle timer. If it is, the client will send the DNS query over the existing connection. If not, the client will establish a new connection and send the query over that connection.

Clients MAY discard their connection to the server before the idle timeout expires. If they do that, they SHOULD close the connection explicitly, using QUIC's CONNECTION_CLOSE mechanisms, and indicating the Application reason "No Error".

Clients and servers MAY close the connection for a variety of other reasons, indicated using QUIC's CONNECTION_CLOSE. Client and servers that send packets over a connection discarded by their peer MAY receive a stateless reset indication. If a connection fails, all queries in progress over the connection MUST be considered failed, and a Server Failure (SERVFAIL, [RFC1035]) SHOULD be notified to the initiator of the transaction.

5.5. Connection Resume and 0-RTT

A stub resolver MAY take advantage of the connection resume mechanisms supported by QUIC transport [I-D.ietf-quic-transport] and QUIC TLS [I-D.ietf-quic-tls]. Stub resolvers SHOULD consider potential privacy issues associated with session resume before deciding to use this mechanism. These privacy issues are detailed in Section 9.2.

When resuming a session, a stub resolver MAY take advantage of the 0-RTT mechanism supported by QUIC. The 0-RTT mechanism MUST NOT be used to send data that is not "replayable" transactions. For example, a stub resolver MAY transmit a Query as 0-RTT, but MUST NOT transmit an Update.

5.6. Message Sizes

DoQ Queries and Responses are sent on QUIC streams, which in theory can carry up to 2^62 bytes. However, DNS messages are restricted in practice to a maximum size of 65535 bytes. This maximum size is enforced by the use of a two-octet message length field in DNS over TCP [RFC1035] and DNS over TLS [RFC7858], and by the definition of the "application/dns-message" for DNS over HTTP [RFC8484]. DoQ enforces the same restriction.

The flow control mechanism of QUIC control how much data can be sent by QUIC nodes at a given time. The initial values of per stream flow control parameters is defined by two transport parameters:

- initial_max_stream_data_bidi_local: when set by the client, specifies the amount of data that servers can send on a "response" stream without waiting for a MAX_STREAM_DATA frame.

- initial_max_stream_data_bidi_remote: when set by the server, specifies the amount of data that clients can send on a "query" stream without waiting for a MAX_STREAM_DATA frame.

For better performance, it is RECOMMENDED that clients and servers set each of these two parameters to a value of 65535 or greater.

The Extension Mechanisms for DNS (EDNS) [RFC6891] allow peers to specify the UDP message size. This parameter is ignored by DoQ. DoQ implementations always assume that the maximum message size is 65535 bytes.

6. Implementation Requirements

6.1. Authentication

For the stub to recursive resolver scenario, the authentication requirements are the same as described in DoT [RFC7858] and "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310]. There is no need to authenticate the client's identity in either scenario.

6.2. Fall Back to Other Protocols on Connection Failure

If the establishment of the DoQ connection fails, clients SHOULD attempt to fall back to DoT and then potentially clear text, as specified in DoT [RFC7858] and "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310], depending on their privacy profile.

DNS clients SHOULD remember server IP addresses that don't support DoQ, including timeouts, connection refusals, and QUIC handshake failures, and not request DoQ from them for a reasonable period (such as one hour per server). DNS clients following an out-of-band key-pinned privacy profile ([RFC7858]) MAY be more aggressive about retrying DoQ connection failures.

6.3. Address Validation

Section 8 of the QUIC transport specification [I-D.ietf-quic-transport] defines Address Validation procedures to avoid servers being used in address amplification attacks. DoQ implementations MUST conform to this specification, which limits the worst case amplification to a factor 3.

DoQ implementations SHOULD consider configuring servers to use the Address Validation using Retry Packets procedure defined in section 8.1.2 of the QUIC transport specification [I-D.ietf-quic-transport]). This procedure imposes a 1-RTT delay for verifying the return routability of the source address of a client, similar to the DNS Cookies mechanism [RFC7873].

DoQ implementations that configure Address Validation using Retry Packets SHOULD implement the Address Validation for Future Connections procedure defined in section 8.1.3 of the QUIC transport specification [I-D.ietf-quic-transport]). This defines how servers can send NEW TOKEN frames to clients after the client address is validated, in order to avoid the 1-RTT penalty during subsequent connections by the client from the same address.

6.4. DNS Message IDs

When sending queries over a QUIC connection, the DNS Message ID MUST be set to zero.

6.5. Padding

There are mechanisms specified for padding individual DNS messages in "The EDNS(0) Padding Option" [RFC7830] and for padding within QUIC packets (see Section 8.6 of the QUIC transport specification [I-D.ietf-quic-transport]).

Implementations SHOULD NOT use DNS options for padding individual DNS messages, because QUIC transport MAY transmit multiple STREAM frames containing separate DNS messages in a single QUIC packet. Instead, implementations SHOULD use QUIC PADDING frames to align the packet length to a small set of fixed sizes, aligned with the recommendations of the "Padding Policies for Extension Mechanisms for DNS (EDNS(0))" [RFC8467].

6.6. Connection Handling

"DNS Transport over TCP - Implementation Requirements" [RFC7766] provides updated guidance on DNS over TCP, some of which is applicable to DoQ. This section attempts to specify which and how those considerations apply to DoQ.

6.6.1. Connection Reuse

Historic implementations of DNS stub resolvers are known to open and close TCP connections for each DNS query. To avoid excess QUIC connections, each with a single query, clients SHOULD reuse a single QUIC connection to the recursive resolver.

In order to achieve performance on par with UDP, DNS clients SHOULD send their queries concurrently over the QUIC streams on a QUIC connection. That is, when a DNS client sends multiple queries to a server over a QUIC connection, it SHOULD NOT wait for an outstanding reply before sending the next query.

6.6.2. Resource Management and Idle Timeout Values

Proper management of established and idle connections is important to the healthy operation of a DNS server. An implementation of DoQ SHOULD follow best practices similar to those specified for DNS over TCP [RFC7766], in particular with regard to:

- Concurrent Connections (Section 6.2.2)

- Security Considerations (Section 10)

  Failure to do so may lead to resource exhaustion and denial of service.

  Clients that want to maintain long duration DoQ connections SHOULD use the idle timeout mechanisms defined in Section 10.2 of the QUIC transport specification [I-D.ietf-quic-transport]. Clients and servers MUST NOT send the edns-tcp-keepalive EDNS(0) Option [RFC7828] in any messages sent on a DoQ connection (because it is specific to the use of TCP/TLS as a transport). If any message sent on a DoQ connection contains an edns-tcp-keepalive EDNS(0) Option, this is a fatal error and the recipient of the defective message MUST forcibly abort the connection immediately.

  This document does not make specific recommendations for timeout values on idle connections. Clients and servers should reuse and/or close connections depending on the level of available resources. Timeouts may be longer during periods of low activity and shorter during periods of high activity.

  Clients that are willing to use QUIC's 0-RTT mechanism can reestablish connections and send transactions on the new connection with minimal delay overhead. These clients MAY chose low values of the idle timer.

6.7. Processing Queries in Parallel

As specified in Section 7 of "DNS Transport over TCP - Implementation Requirements" [RFC7766], resolvers are RECOMMENDED to support the preparing of responses in parallel and sending them out of order. In DoQ, they do that by sending responses on their specific stream as soon as possible, without waiting for availability of responses for previously opened streams.

6.8. Flow Control Mechanisms

Servers and Clients manage flow control as specified in QUIC.

Servers MAY use the "maximum stream ID" option of the QUIC transport to limit the number of streams opened by the client. This mechanism will effectively limit the number of DNS queries that a client can send on a single DoQ connection.

7. Implementation Status

(THIS SECTION TO BE REMOVED BEFORE PUBLICATION) This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].

1. AdGuard launched a DoQ recursive resolver service in December 2020. They have released a suite of open source tools that support DoQ:

    1. AdGuard C++ DNS libraries [1] A DNS proxy library that supports all existing DNS protocols including DNS-over-TLS, DNS-over-HTTPS, DNSCrypt and DNS-over-QUIC (experimental).

    2. DNS Proxy [2] A simple DNS proxy server that supports all existing DNS protocols including DNS-over-TLS, DNS-over-HTTPS, DNSCrypt, and DNS-over-QUIC. Moreover, it can work as a DNS-over-HTTPS, DNS-over-TLS or DNS-over-QUIC server.

    3. CoreDNS fork for AdGuard DNS [3] Includes DNS-over-QUIC server-side support.

    4. dnslookup [4] Simple command line utility to make DNS lookups. Supports all known DNS protocols: plain DNS, DoH, DoT, DoQ, DNSCrypt.

2. Quicdoq [5] Quicdoq is a simple open source implementation of DNS over Quic. It is written in C, based on Picoquic [6].

3. Flamethrower [7] is an open source DNS performance and functional testing utility written in C++ that has an experimental implementation of DoQ.

4. aioquic [8] is an implementation of QUIC in Python. It includes example client and server for DNS over QUIC.

7.1. Performance Measurements

To our knowledge, no benchmarking studies comparing DoT, DoH and DoQ are published yet. However anecdotal evidence from the AdGuard DoQ recursive resolver deployment [9] indicates that it performs well compared to the other encrypted protocols, particularly in mobile environments. Reasons given for this include that DoQ

- Uses less bandwidth due to a more efficient handshake (and due to less per message overhead when compared to DoH).

- Performs better in mobile environments due to the increased resilience to packet loss

- Can maintain connections as users move between mobile networks via its connection management

8. Security Considerations

The security considerations of DoQ should be comparable to those of DoT [RFC7858].

9. Privacy Considerations

DoQ is specifically designed to protect the DNS traffic between stub and resolver from observations by third parties, and thus protect the privacy of queries sent by the stub. However, the recursive resolver has full visibility of the stub's traffic, and could be used as an observation point, as discussed in the revision of "DNS Privacy Considerations" [I-D.ietf-dprive-rfc7626-bis]. These considerations do not differ between DoT and DoQ and are not discussed further here.

QUIC incorporates the mechanisms of TLS 1.3 [RFC8446] and this enables QUIC transmission of "0-RTT" data. This can provide interesting latency gains, but it raises two concerns:

1. Adversaries could replay the 0-RTT data and infer its content from the behavior of the receiving server.

2. The 0-RTT mechanism relies on TLS resume, which can provide linkability between successive client sessions.

These issues are developed in Section 9.1 and Section 9.2.

9.1. Privacy Issues With 0-RTT data

The 0-RTT data can be replayed by adversaries. That data may trigger queries by a recursive resolver to authoritative resolvers. Adversaries may be able to pick a time at which the recursive resolver outgoing traffic is observable, and thus find out what name was queried for in the 0-RTT data.

This risk is in fact a subset of the general problem of observing the behavior of the recursive resolver discussed in "DNS Privacy Considerations" [RFC7626]. The attack is partially mitigated by reducing the observability of this traffic. However, the risk is amplified for 0-RTT data, because the attacker might replay it at chosen times, several times.

The recommendation for TLS 1.3 [RFC8446] is that the capability to use 0-RTT data should be turned off by default, and only enabled if the user clearly understands the associated risks.

QUESTION: Should 0-RTT only be used with Opportunistic profiles (i.e. disabled by default for Strict only)?

9.2. Privacy Issues With Session Resume

The QUIC session resume mechanism reduces the cost of re-establishing sessions and enables 0-RTT data. There is a linkability issue associated with session resume, if the same resume token is used several times, but this risk is mitigated by the mechanisms incorporated in QUIC and in TLS 1.3. With these mechanisms, clients and servers can cooperate to avoid linkability by third parties. However, the server will always be able to link the resumed session to the initial session. This creates a virtual long duration session. The series of queries in that session can be used by the server to identify the client.

Enabling the server to link client sessions through session resume is probably not a large additional risk if the client's connectivity did not change between the sessions, since the two sessions can probably be correlated by comparing the IP addresses. On the other hand, if the addresses did change, the client SHOULD consider whether the linkability risk exceeds the performance benefits. This evaluation will obviously depend on the level of trust between stub and recursive.

9.3. Traffic Analysis

Even though QUIC packets are encrypted, adversaries can gain information from observing packet lengths, in both queries and responses, as well as packet timing. Many DNS requests are emitted by web browsers. Loading a specific web page may require resolving dozen of DNS names. If an application adopts a simple mapping of one query or response per packet, or "one QUIC STREAM frame per packet", then the succession of packet lengths may provide enough information to identify the requested site.

Implementations SHOULD use the mechanisms defined in Section 6.5 to mitigate this attack.

10. IANA Considerations

10.1. Registration of DoQ Identification String

This document creates a new registration for the identification of DoQ in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry [RFC7301].

The "doq" string identifies DoQ:

Protocol: DoQ

Identification Sequence: 0x64 0x6F 0x71 ("doq")

Specification: This document

10.2. Reservation of Dedicated Port

IANA is required to add the following value to the "Service Name and Transport Protocol Port Number Registry" in the System Range. The registry for that range requires IETF Review or IESG Approval [RFC6335], and such a review was requested using the early allocation process [RFC7120] for the well-known UDP port in this document. Since port 853 is reserved for 'DNS query-response protocol run over TLS' consideration is requested for reserving port 8853 for 'DNS query-response
protocol run over QUIC'.

Service Name	dns-over-quic
Port Number	8853
Transport Protocol(s)	UDP
Assignee	IESG
Contact	IETF Chair
Description	DNS query-response protocol run over QUIC
Reference	This document

10.2.1. Port number 8853 for experimentations

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Early experiments MAY use port 8853. This port is marked in the IANA registry as unassigned.

(Note that prior to version -02 of this draft, experiments were directed to use port 784.)

11. Acknowledgements

This document liberally borrows text from the HTTP-3 specification [I-D.ietf-quic-http] edited by Mike Bishop, and from the DoT specification [RFC7858] authored by Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman.

The privacy issue with 0-RTT data and session resume were analyzed by Daniel Kahn Gillmor (DKG) in a message to the IETF "DPRIVE" working group [DNS0RTT].

Thanks to Tony Finch for an extensive review of the initial version of this draft. Reviews by Paul Hoffman and interoperability tests conducted by Stephane Bortzmeyer helped improve the definition of the protocol.

12. References

12.1. Normative References

[I-D.ietf-dnsop-terminology-ter]
Hoffman, P., "Terminology for DNS Transports and Location", draft-ietf-dnsop-terminology-ter-02 (work in progress), August 2020.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using TLS to Secure QUIC", draft-ietf-quic-tls-34 (work in progress), January 2021.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-34 (work in progress), January 2021.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

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

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/info/rfc7766>.

[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, <https://www.rfc-editor.org/info/rfc7858>.

[RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <https://www.rfc-editor.org/info/rfc7873>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

12.2. Informative References

[DNS0RTT] Kahn Gillmor, D., "DNS + 0-RTT", Message to DNS-Privacy WG mailing list, April 2016, <https://www.ietf.org/mail-archive/web/dns-privacy/current/msg01276.html>.

[I-D.ietf-dprive-phase2-requirements] Livingood, J., Mayrhofer, A., and B. Overeinder, "DNS Privacy Requirements for Exchanges between Recursive Resolvers and Authoritative Servers", draft-ietf-dprive-phase2-requirements-02 (work in progress), November 2020.

[I-D.ietf-dprive-rfc7626-bis] Wicinski, T., "DNS Privacy Considerations", draft-ietf-dprive-rfc7626-bis-08 (work in progress), October 2020.

[I-D.ietf-dprive-xfr-over-tls]
Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer-over-TLS", draft-ietf-dprive-xfr-over-tls-05 (work in progress), January 2021.

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-33 (work in progress), December 2020.

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-34 (work in progress), January 2021.

[RFC1885] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)", RFC 1885, DOI 10.17487/RFC1885, December 1995, <https://www.rfc-editor.org/info/rfc1885>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <https://www.rfc-editor.org/info/rfc7626>.

[RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", RFC 7828, DOI 10.17487/RFC7828, April 2016, <https://www.rfc-editor.org/info/rfc7828>.

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

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/info/rfc8094>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <https://www.rfc-editor.org/info/rfc8467>.

[RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <https://www.rfc-editor.org/info/rfc8490>.

12.3. URIs

[1] https://github.com/AdguardTeam/DnsLibs

[2] https://github.com/AdguardTeam/dnsproxy

[3] https://github.com/AdguardTeam/coredns

[4] https://github.com/ameshkov/dnslookup

[5] https://github.com/private-octopus/quicdoq

[6] https://github.com/private-octopus/picoquic

[7] https://github.com/DNS-OARC/flamethrower/tree/dns-over-quic

[8] https://github.com/aiortc/aioquic

[9] https://adguard.com/en/blog/dns-over-quic.html

Appendix A. Supporting AXFR

This draft version makes no attempt to support AXFR or IXFR queries. As defined in [RFC5936], the server responds to AXFR queries with a series of DNS response messages where

"... the first message MUST begin with the SOA resource record of the zone, and the last message MUST conclude with the same SOA resource record."

and the QDCOUNT:

- MUST be 1 in the first message;
- MUST be 0 or 1 in all following messages;
- MUST be 1 if RCODE indicates an error

When the DNS protocol is carried over TCP or TLS, these messages are carried over a single byte stream and each of them is preceded by a 16 bit length field. The encapsulation currently defined in this draft does not include a length field and assumes exactly one response message for each query.

Note that since IXFR can fall back to an AXFR-like response if the server is not able to send an incremental change, this discussion also applies to those AXFR-like responses returned to an IXFR request in that scenario.

There are two plausible ways to carry the series of AXFR responses in QUIC: keep the current format and use a separate QUIC stream for each response; or, relax the restriction of having just one response per QUIC stream. This second option is much simpler to engineer. It will not require complex methods to correlate different streams, and it will ensure that the responses in the series are delivered in the intended order. However, it requires parsing the response stream to extract separate responses. The practical requirement would be that the content of the QUIC stream be exactly the same as the content of a TCP connection that would manage exactly one query. The main difference with the current proposal would be to insert a length field before each response. So we would get:

- For a query: open a bidirectional stream, send the query encoded as { 16 bit length, DNS query }, mark this stream direction as finished.
- For most responses: send the single response message encoded as { 16 bit length, DNS response }, mark this stream direction as finished.
- For a response to an AXFR query: send a series of response messages encoded as { 16 bit length, DNS response }, using the QDCOUNT convention as specified in [RFC5936], mark this stream direction as finished when the entire series is sent.

This adds a length field that is not in the current draft, which breaks compatibility with the previous versions. Draft versions are identified by draft version specific ALPN, which makes this change manageable. However, the authors would like to get feedback from developers before making this change.

The change will also add new error conditions: if the stream FIN happens before the bytes specified in the message length field are sent; if the client expects a single response message and several are sent; and, if the client expects AXFR responses but does not receive the expected pattern of QDCOUNT flagged messages.

Authors' Addresses

Christian Huitema
Private Octopus Inc.
427 Golfcourse Rd
Friday Harbor WA 98250
U.S.A

Email: huitema@huitema.net

Allison Mankin
Salesforce

Email: amankin@salesforce.com

Sara Dickinson
Sinodun IT
Oxford Science Park
Oxford OX4 4GA
U.K.

Email: sara@sinodun.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 3 October 2021

P. Hoffman
ICANN
P. van Dijk
PowerDNS

1 April 2021

Recursive to Authoritative DNS with Unauthenticated Encryption
draft-ietf-dprive-opportunistic-adotq-02

Abstract

This document describes a use case and a method for a DNS recursive resolver to use unauthenticated encryption when communicating with authoritative servers. The motivating use case for this method is that more encryption on the Internet is better, and some resolver operators believe that unauthenticated encryption is better than no encryption at all. The method described here is optional for both the recursive resolver and the authoritative server. This method supports unauthenticated encryption using the same mechanism for discovery of encryption support for the server as [I-D.rescorla-dprive-adox-latest].

NOTE: The file name for this draft, draft-ietf-dprive-opportunistic-adotq, is now incorrect. This draft only covers unauthenticated encryption, not opportunistic encryption.

Table of Contents

1. Introduction

A recursive resolver using traditional DNS over port 53 may wish instead to use encrypted communication with authoritative servers in order to limit snooping of its DNS traffic by passive or on-path attackers. The recursive resolver can use unauthenticated encryption (defined in [RFC7435]) to achieve this goal.

This document describes the use case for unauthenticated encryption in recursive resolvers in Section 1.1. The encryption method with authoritative servers can be DNS-over-TLS [RFC7858] (DoT), DNS-over-HTTPS [RFC8484] (DoH), and/or DNS-over-QUIC [I-D.ietf-dprive-dnsoquic] (DoQ), as described in Section 3.

The document also describes a discovery method that shows if an authoritative server supports encryption in Section 2.

See [I-D.rescorla-dprive-adox-latest] for a description of the use case and a proposed mechanism for fully-authenticated encryption.

NOTE: The draft uses the SVCB record as a discovery mechanism for encryption by a particular authoritative server. Any record type that can show multiple types of encryption (currently DoT, DoH, and DoQ) can be used for discovery. Thus, this record type might change in the future, depending on the discussion in the DPRIVE WG.

1.1. Use Case for Unauthenticated Encryption

The use case in this document for unauthenticated encryption is recursive resolver operators who are happy to use encryption with authoritative servers if doing so doesn't significantly slow down getting answers, and authoritative server operators that are happy to use encryption with recursive resolvers if it doesn't cost much. In this use case, resolvers do not want to return an error for requests that were sent over an encrypted channel if they would have been able to give a correct answer using unencrypted transport.

Resolvers and authoritative servers understand that using encryption costs something, but are willing to absorb the costs for the benefit of more Internet traffic being encrypted. The extra costs (compared to using traditional DNS on port 53) include:

* Extra round trips to establish TCP for every session (but not necessarily for every query)

* Extra round trips for TLS establishment

* Greater CPU use for TLS establishment

* Greater CPU use for encryption after TLS establishment

* Greater memory use for holding TLS state

This use case is not expected to apply to all resolvers or authoritative servers. For example, according to [RSO_STATEMENT], some root server operators do not want to be the early adopters for DNS with encryption. The protocol in this document explicitly allows authoritative servers to signal when they are ready to begin offering DNS with encryption.

1.2. Summary of Protocol

This summary gives an overview of how the parts of the protocol work together.

* The resolver discovers whether any authoritative server of interest supports DNS with encryption by querying for the SVCB records [I-D.ietf-dnsop-svcb-https]. As described in [I-D.schwartz-svcb-dns], SVCB records can indicate that a server supports encrypted transport of DNS queries.

  NOTE: In this document, the term "SVCB record" is used _only_ for SVCB records that indicate encryption as described in [I-D.schwartz-svcb-dns]. SVCB records that do not have these indicators in the RDATA are not included in the term "SVCB record" in this document.

* The resolver uses any authoritative server with a SVCB record that indicates encryption to perform unauthenticated encryption.

* The resolver does not fail to set up encryption if the authentication in the TLS session fails.

1.3. Definitions

The terms "recursive resolver", "authoritative server", and "classic DNS" are defined in [I-D.ietf-dnsop-rfc8499bis].

"DNS with encryption" means transport of DNS over any of DoT, DoH, or DoQ. A server that supports DNS with encryption supports transport over one or more of DoT, DoH, or DoQ.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Discovering Whether an Authoritative Server Uses Encryption

A recursive resolver discovers whether an authoritative server supports DNS with encryption by looking for a cached SVCB record for the name of the authoritative server (with "_dns" prefix) with a positive answer. A cached SVCB record with a negative answer indicates that the authoritative server does not support any encrypted transport. Positive and negative responses for SVCB queries are cached the same way as for all other DNS resource records.

See [I-D.rescorla-dprive-adox-latest] for examples of querying for NS records and for SVCB records, and the interpretation of positive answers.

If the cache has no positive or negative answers for any SVCB record for any of a zone's authoritative servers, the resolver MAY send queries for the SVCB records for some or all of the zone's authoritative servers and wait for a positive response so that the resolver can use DNS with encryption for the original query. In this situation, the resolver MAY instead just use classic DNS for the original query but simultaneously queue queries for the SVCB records for some or all of the zone's authoritative servers so that future queries might be able to use DNS with encryption.

Discovery using SVCB records differs between resolvers using unauthenticated encryption and those using fully-authenticated encryption (described in [I-D.rescorla-dprive-adox-latest]). If the resolver is using unauthenticated encryption, the SVCB records do not need to be DNSSEC-signed.

DNSSEC validation of SVCB RRsets used strictly for this discovery mechanism is not mandated.

As described in [I-D.rescorla-dprive-adox-latest], these records may be in the resolver's cache because they came in the Additional section of a query for the NS records of a zone. This document does not rely on that feature being standardized or operationally present to work.

Because some authoritative servers or middleboxes are misconfigured, requests for unknown RRtypes might be ignored by them. Resolvers should be ready to deal with timeouts or other bad responses to their SVCB queries.

3. Resolving with Encryption

A resolver following this protocol MUST use SVCB records in its cache to decide whether to use classic DNS or encryption to contact authoritative servers for a zone. If any of the SVCB records in the cache for the authoritative servers for a zone are positive responses, the resolver uses any of those servers for encryption. A resolver MUST NOT attempt encryption for a server that has a negative response in its cache for the associated SVCB record.

If all of the SVCB records for the authoritative servers in the cache for a zone are negative responses, the resolver MUST use classic (unencrypted) DNS instead of encryption. Similarly, if none of the SVCB records for the authoritative servers in the cache have information about encrypted services as described in [I-D.schwartz-svcb-dns], the resolver MUST use classic (unencrypted) DNS instead of encryption.

If there are any SVCB records in the cache for the authoritative servers for a zone with a positive response, the resolver MUST try each indicated authoritative server using DNS with encryption until it successfully sets up a connection. The resolver only attempts to use the encrypted transports that are in the associated SVCB record for the authoritative server. Reasons for TLS failures are listed in Section 3.1.

After a DNS with encryption session is set up, the resolver uses that authoritative server for whatever query about the zone it was going to send. If a resolver cannot set up a DNS with encryption session with any of the authoritative servers, it MUST attempt to perform the resolution over classic (unencrypted) DNS as it would have without encryption.

A resolver SHOULD keep a DNS with encryption session to a particular server open if it expects to send additional queries to that server in a short period of time. If the server closes the DNS with encryption session, the resolver can possibly re-establish a DNS with encryption session using encrypted session resumption. [RFC7766] says "both clients and servers SHOULD support connection reuse" for TCP connections, and that advice could apply as well for DNS with encryption even though DNS with encryption has greater overhead for saving state.

Privacy-oriented resolvers (defined in [RFC8932]) following this protocol MUST NOT indicate that they are using encryption because this protocol is susceptible to on-path attacks.

3.1. Resolver Session Failures

The following are the reasons that a DNS with encryption session might fail to be set up:

* The resolver receives a TCP RST response

* The resolver does not receive replies to TCP or TLS setup (such as getting the TCP SYN message, the first TLS message, or completing TLS handshakes)

* The TLS handshake gets a definitive failure

* The encrypted session fails for reasons other than for authentication, such as incorrect algorithm choices or TLS record failures

4. Serving with Encryption

An operator of an authoritative server following this protocol SHOULD publish SVCB records as described in Section 2. If they cannot publish such records, the security properties of their authoritative servers will not be found. If an operator wants to test serving using encryption, they can publish SVCB records with short TTLs and then stop serving with encryption after removing the SVCB records and waiting for the TTLs to expire.

An operator of authoritative servers for a zone that is following this protocol MAY support encryption towards any IP address on which it offers service for classic DNS on port 53. It is acceptable for such an operator to only offer encryption on some of the named authoritative servers, such as when the operator is determining how far to roll out encrypted service.

A server MAY close an encrypted connection at any time. For example, it can close the session if it has not received a DNS query in a defined length of time. The server MAY close an encrypted session after it sends a DNS response; however, it might also want to keep the session open waiting for another DNS query from the resolver. [RFC7766] says "both clients and servers SHOULD support connection reuse" for TCP connections, and that advice could apply as well for DNS with encryption even though DNS with encryption has greater overhead for saving state.

5. Resolvers Reporting Errors to Authoritative Servers

Resolvers should have a method of telling authoritative servers that there are problems with the encrypted service they are offering. There is a proposal that the DNSOP Working Group adopt [I-D.arends-dns-error-reporting], which would enable such reporting.

((Clearly, more will need to go here.))

6. IANA Considerations

((Update registration for TCP/853 to also include ADoT))

((Maybe other updates for DoH and DoQ))

7. Security Considerations

The method described in this document explicitly allows a resolver to perform DNS communications over traditional unencrypted, unauthenticated DNS on port 53, if it cannot find an authoritative server that advertises that it supports encryption. The method described in this document explicitly allows a resolver using encryption to choose to allow unauthenticated encryption. In either of these cases, the resulting communication will be susceptible to obvious and well-understood attacks from an attacker in the path of the communications.

An authoritative server that wants to only serve data to resolvers that using fully-authenticated encryption as described in [I-D.rescorla-dprive-adox-latest] cannot differentiate between those resolvers and resolvers using the mechanisms described in this document.

8. Acknowledgements

Puneet Sood contributed many ideas to early drafts of this document.

The DPRIVE Working Group has contributed many ideas that keep shifting the focus and content of this document.

9. References

9.1. Normative References

[I-D.ietf-dnsop-rfc8499bis]
Hoffman, P. and K. Fujiwara, "DNS Terminology", Work in Progress, Internet-Draft, draft-ietf-dnsop-rfc8499bis-01, 20 November 2020, <https://www.ietf.org/archive/id/draft-ietf-dnsop-rfc8499bis-01.txt>.

[I-D.ietf-dnsop-svcb-https]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-04, 17 March 2021, <https://www.ietf.org/archive/id/draft-ietf-dnsop-svcb-https-04.txt>.

[I-D.rescorla-dprive-adox-latest]
Pauly, T., Rescorla, E., Schinazi, D., and C. A. Wood, "Signaling Authoritative DNS Encryption", Work in Progress, Internet-Draft, draft-rescorla-dprive-adox-latest-00, 26 February 2021, <https://www.ietf.org/archive/id/draft-rescorla-dprive-adox-latest-00.txt>.

[I-D.schwartz-svcb-dns]
Schwartz, B., "Service Binding Mapping for DNS Servers", Work in Progress, Internet-Draft, draft-schwartz-svcb-dns-02, 17 February 2021, <https://www.ietf.org/archive/id/draft-schwartz-svcb-dns-02.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/info/rfc7766>.

[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, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

9.2. Informative References

[I-D.arends-dns-error-reporting]
Arends, R. and M. Larson, "DNS Error Reporting", Work in Progress, Internet-Draft, draft-arends-dns-error-reporting-00, 30 October 2020, <https://www.ietf.org/archive/id/draft-arends-dns-error-reporting-00.txt>.

[I-D.ietf-dprive-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-ietf-dprive-dnsoquic-02, 22 February 2021, <https://www.ietf.org/archive/id/draft-ietf-dprive-dnsoquic-02.txt>.

[RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <https://www.rfc-editor.org/info/rfc8932>.

[RSO_STATEMENT]
"Statement on DNS Encryption", 2021, <https://root-servers.org/media/news/Statement_on_DNS_Encryption.pdf>.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Peter van Dijk
PowerDNS

Email: peter.van.dijk@powerdns.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 September 2021

E. Kinnear
Apple Inc.
P. McManus
Fastly
T. Pauly
Apple Inc.
T. Verma
C.A. Wood
Cloudflare

8 March 2021

Oblivious DNS Over HTTPS
draft-pauly-dprive-oblivious-doh-06

Abstract

This document describes an extension to DNS Over HTTPS (DoH) that allows hiding client IP addresses via proxying encrypted DNS transactions. This improves privacy of DNS operations by not allowing any one server entity to be aware of both the client IP address and the content of DNS queries and answers.

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 https://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 9 September 2021.

Table of Contents

1. Introduction

DNS Over HTTPS (DoH) [RFC8484] defines a mechanism to allow DNS messages to be transmitted in encrypted HTTP messages. This provides improved confidentiality and authentication for DNS interactions in various circumstances.

While DoH can prevent eavesdroppers from directly reading the contents of DNS exchanges, clients cannot send DNS queries and receive answers from servers without revealing their local IP address, and thus information about the identity or location of the client.

Proposals such as Oblivious DNS ([I-D.annee-dprive-oblivious-dns]) increase privacy by ensuring no single DNS server is aware of both the client IP address and the message contents.

This document defines Oblivious DoH, an extension to DoH that permits proxied resolution, in which DNS messages are encrypted so that no DoH server can independently read both the client IP address and the DNS message contents.

This mechanism is intended to be used as one option for resolving privacy-sensitive content in the broader context of Adaptive DNS [I-D.pauly-dprive-adaptive-dns-privacy].

1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

This document defines the following terms:

Oblivious Server: A DoH server that acts as either an Oblivious Proxy or Oblivious Target.

Oblivious Proxy: An Oblivious Server that proxies encrypted DNS queries and responses between a client and an Oblivious Target.

Oblivious Target: An Oblivious Server that receives and decrypts encrypted client DNS queries from an Oblivious Proxy, and returns encrypted DNS responses via that same Proxy. In order to provide DNS responses, the Target can be a DNS resolver, be co-located with a resolver, or forward to a resolver.

Throughout the rest of this document, we use the terms Proxy and Target to refer to an Oblivious Proxy and Oblivious Target, respectively.

3. Deployment Requirements

Oblivious DoH requires, at a minimum:

* Two Oblivious Servers, where one can act as a Proxy, and the other can act as a Target.

* Public keys for encrypting DNS queries that are passed from a client through a Proxy to a Target (Section 6). These keys guarantee that only the intended Target can decrypt client queries.

The mechanism for discovering and provisioning the DoH URI Templates and public keys is via parameters added to DNS resource records. The mechanism for discovering the public key is described in Section 5. The mechanism for discovering a DoH URI Template is described in [I-D.pauly-add-resolver-discovery].

4. HTTP Exchange

Unlike direct resolution, oblivious hostname resolution over DoH involves three parties:

1. The Client, which generates queries.

2. The Proxy, which receives encrypted queries from the client and passes them on to a Target.

3. The Target, which receives proxied queries from the client via the Proxy and produces proxied answers.

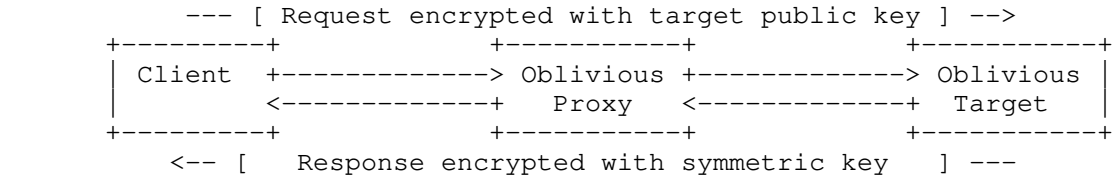

Figure 1: Obvlivious DoH Exchange

4.1. HTTP Request

Oblivious DoH queries are created by the Client, and sent to the Proxy. Requests to the Proxy indicate which DoH server to use as a Target by specifying two variables: "targethost", which indicates the host name of the Target server, and "targetpath", which indicates the path on which the Target's DoH server is running. See Section 4.2 for an example request.

Oblivious DoH messages have no cache value since both requests and responses are encrypted using ephemeral key material. Clients SHOULD prefer using HTTP methods and headers that will prevent unhelpful cache storage of these exchanges (i.e., preferring POST instead of GET).

Clients MUST set the HTTP Content-Type header to "application/oblivious-dns-message" to indicate that this request is an Oblivious DoH query intended for proxying. Clients also SHOULD set this same value for the HTTP Accept header.

Proxies must check that client requests are correctly encoded, and MUST return a 4xx (Client Error) if the check fails, along with the Proxy-Status response header with an "error" parameter of type "http_request_error" [I-D.ietf-httpbis-proxy-status]. A correctly encoded request has the HTTP Content-Type header "application/oblivious-dns-message", and HTTP method POST. If the proxy does not operate as a target, then the request must additionally contain "targethost" and "targetpath" variables.

Upon receiving a request that contains a "application/oblivious-dns-message" Content-Type, the DoH server looks for the "targethost" and "targetpath" variables. If the variables are not present, then it is the target of the query, and it can decrypt the query (Section 7). If the variables are present, then the DoH server is acting as a Proxy. If it is a proxy, it is expected to send the request on to the Target using the URI template constructed as "https://targethost/targetpath".

Note that "targethost" may contain a port. Proxies MAY choose to not forward connections to non-standard ports. In such cases, proxies MUST return a 4xx (Client Error) response to the client request, along with Proxy-Status response header with an "error" parameter of type "http_request_error".

If the proxy cannot establish a connection to "targethost", it MUST return a 502 (Bad Gateway) response to the client request, along with Proxy-Status response header with an "error" parameter whose type indicates the reason. For example, if DNS resolution fails, the error type might be "dns_timeout", whereas if the TLS connection failed the error type might be "tls_protocol_error". Proxies SHOULD choose an error type that best captures the connection failure.

4.2. HTTP Request Example

The following example shows how a client requests that a Proxy, "dnsproxy.example.net", forwards an encrypted message to "dnstarget.example.net". The URI template for the Proxy is "https://dnsproxy.example.net/dns-query{?targethost,targetpath}". The URI template for the Target is "https://dnstarget.example.net/dns-query".

```
:method = POST
:scheme = https
:authority = dnsproxy.example.net
:path = /dns-query?targethost=dnstarget.example.net&targetpath=/dns-query
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

The Proxy then sends the following request on to the Target:

```
:method = POST
:scheme = https
:authority = dnstarget.example.net
:path = /dns-query
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

4.3. HTTP Response

The response to an Oblivious DoH query is generated by the Target. It MUST set the Content-Type HTTP header to "application/oblivious-dns-message" for all successful responses. The body of the response contains an encrypted DNS message; see Section 7.

The response from a Target MUST set the Content-Type HTTP header to "application/oblivious-dns-message" which MUST be forwarded by the Proxy to the Client. A Client MUST only consider a response which contains the Content-Type header in the response before processing the payload. A response without the appropriate header MUST be treated as an error and be handled appropriately. All other aspects of the HTTP response and error handling are inherited from standard DoH.

Proxies MUST forward responses unmodified to clients. Specifically, the HTTP status code generated by Targets must be forwarded to clients unmodified. If a proxy receives a successful response from a target without the "application/oblivious-dns-message" HTTP Content-Type header, it MUST return a 502 (Bad Gateway) response to the client request, along with Proxy-Status response header with an "error" parameter of type "http_protocol_error".

4.4. HTTP Response Example

The following example shows a 2xx (Successful) response that can be sent from a Target to a client via a Proxy.

```
:status = 200
content-type = application/oblivious-dns-message
content-length = 154

<Bytes containing the encrypted payload for an Oblivious DNS response>
```

Requests that cannot be processed by the target result in 4xx (Client Error) responses. If the target and client keys do not match, it is an authorization failure (HTTP status code 401; see Section 3.1 of [RFC7235]). Otherwise, if the client's request is invalid, such as in the case of decryption failure, wrong message type, or deserialization failure, this is a bad request (HTTP status code 400; see Section 6.5.1 of [RFC7231]).

Even in case of DNS responses indicating failure, such as SERVFAIL or NXDOMAIN, a successful HTTP response with a 2xx status code is used as long as the DNS response is valid. This is similar to how DoH [RFC8484] handles HTTP response codes.

In case of server error, the usual HTTP status code 500 (see Section 6.6.1 of [RFC7231]) applies.

4.5. HTTP Metadata

Proxies forward requests and responses between clients and targets as specified in Section 4.1. Metadata sent with these messages may inadvertently weaken or remove Oblivious DoH privacy properties. Proxies MUST NOT send any client-identifying information about clients to targets, such as "Forwarded" HTTP headers [RFC7239]. Additionally, clients MUST NOT include any private state in requests to proxies, such as HTTP cookies.

5. Configuration and Public Key Discovery

In order to use a DoH server as a Target, the client must know a public key to use for encrypting its queries. This document specifies one discovery mechanism for public keys using the SVCB or HTTPSSVC record type ([I-D.ietf-dnsop-svcb-https]) for a name owned by the server.

The Service Binding key name is "odohconfig" (Section 12). If present, this key/value pair contains the public key to use when encrypting Oblivious DoH messages that will be targeted at a DoH server. The format of the key is defined in (Section 6).

Clients that use this discovery mechansim MUST only use keys that were retrieved from records protected by DNSSEC [RFC4033] to encrypt messages to a Target.

Servers SHOULD rotate public keys regularly. It is RECOMMENDED that servers rotate keys every day. Shorter rotation windows reduce the anonymity set of clients that might use the public key, whereas longer rotation windows widen the timeframe of possible compromise.

6. Configuration and Public Key Format

An Oblivious DNS public key configuration is a structure encoded, using TLS-style encoding [RFC8446], as follows:

```
struct {
   uint16 kem_id;
   uint16 kdf_id;
   uint16 aead_id;
   opaque public_key<1..2^16-1>;
} ObliviousDoHConfigContents;

struct {
   uint16 version;
   uint16 length;
   select (ObliviousDoHConfig.version) {
      case 0xff06: ObliviousDoHConfigContents contents;
   }
} ObliviousDoHConfig;

ObliviousDoHConfig ObliviousDoHConfigs<1..2^16-1>;
```

The "ObliviousDoHConfigs" structure contains one or more "ObliviousDoHConfig" structures in decreasing order of preference. This allows a server to support multiple versions of Oblivious DoH and multiple sets of Oblivious DoH parameters.

An "ObliviousDoHConfig" contains a versioned representation of an Oblivious DoH configuration, with the following fields.

version The version of Oblivious DoH for which this configuration is used. Clients MUST ignore any "ObliviousDoHConfig" structure with a version they do not support. The version of Oblivious DoH specified in this document is "0xff06".

length The length, in bytes, of the next field.

contents An opaque byte string whose contents depend on the version. For this specification, the contents are an "ObliviousDoHConfigContents" structure.

An "ObliviousDoHConfigContents" contains the information needed to encrypt a message under "ObliviousDoHConfigContents.public_key" such that only the owner of the corresponding private key can decrypt the message. The values for "ObliviousDoHConfigContents.kem_id", "ObliviousDoHConfigContents.kdf_id", and "ObliviousDoHConfigContents.aead_id" are described in [I-D.irtf-cfrg-hpke] Section 7. The fields in this structure are as follows:

kem_id The HPKE KEM identifier corresponding to "public_key". Clients MUST ignore any "ObliviousDoHConfig" structure with a key using a KEM they do not support.

kdf_id The HPKE KDF identifier corresponding to "public_key". Clients MUST ignore any "ObliviousDoHConfig" structure with a key using a KDF they do not support.

aead_id The HPKE AEAD identifier corresponding to "public_key". Clients MUST ignore any "ObliviousDoHConfig" structure with a key using an AEAD they do not support.

public_key The HPKE public key used by the client to encrypt Oblivious DoH queries.

7. Protocol Encoding

7.1. Message Format

There are two types of Oblivious DoH messages: Queries (0x01) and Responses (0x02). Both messages carry the following information:

1. A DNS message, which is either a Query or Response, depending on context.

2. Padding of arbitrary length which MUST contain all zeros.

They are encoded using the following structure:

```
struct {
   opaque dns_message<1..2^16-1>;
   opaque padding<0..2^16-1>;
} ObliviousDoHMessagePlaintext;
```

Both Query and Response messages use the "ObliviousDoHMessagePlaintext" format.

```
ObliviousDoHMessagePlaintext ObliviousDoHQuery;
ObliviousDoHMessagePlaintext ObliviousDoHResponse;
```

An encrypted "ObliviousDoHMessagePlaintext" is carried in a "ObliviousDoHMessage" message, encoded as follows:

```
struct {
   uint8  message_type;
   opaque key_id<0..2^16-1>;
   opaque encrypted_message<1..2^16-1>;
} ObliviousDoHMessage;
```

The "ObliviousDoHMessage" structure contains the following fields:

message_type A one-byte identifier for the type of message. Query messages use "message_type" 0x01, and Response messages use "message_type" 0x02.

key_id The identifier of the corresponding "ObliviousDoHConfigContents" key. This is computed as "Expand(Extract("", config), "odoh key id", Nh)", where "config" is the ObliviousDoHConfigContents structure and "Extract", "Expand", and "Nh" are as specified by the HPKE cipher suite KDF corresponding to "config.kdf_id".

encrypted_message An encrypted message for the Oblivious Target (for Query messages) or client (for Response messages). Implementations MAY enforce limits on the size of this field depending on the size of plaintext DNS messages. (DNS queries, for example, will not reach the size limit of 2^16-1 in practice.)

The contents of "ObliviousDoHMessage.encrypted_message" depend on "ObliviousDoHMessage.message_type". In particular, "ObliviousDoHMessage.encrypted_message" is an encryption of a "ObliviousDoHQuery" if the message is a Query, and "ObliviousDoHResponse" if the message is a Response.

7.2. Encryption and Decryption Routines

Clients use the following utility functions for encrypting a Query and decrypting a Response as described in Section 8.

encrypt_query_body: Encrypt an Oblivious DoH query.

```
def encrypt_query_body(pkR, key_id, Q_plain):
  enc, context = SetupBaseS(pkR, "odoh query")
  aad = 0x01 || len(key_id) || key_id
  ct = context.Seal(aad, Q_plain)
  Q_encrypted = enc || ct
  return Q_encrypted
```

decrypt_response_body: Decrypt an Oblivious DoH response.

```
def decrypt_response_body(context, Q_plain, R_encrypted, response_nonce):
  aead_key, aead_nonce = derive_secrets(context, Q_plain, response_nonce)
  aad = 0x02 || len(response_nonce) || response_nonce
  R_plain, error = Open(key, nonce, aad, R_encrypted)
  return R_plain, error
```

The "derive_secrets" function is described below.

Targets use the following utility functions in processing queries and producing responses as described in Section 9.

setup_query_context: Set up an HPKE context used for decrypting an Oblivious DoH query.

```
def setup_query_context(skR, key_id, Q_encrypted):
  enc || ct = Q_encrypted
  context = SetupBaseR(enc, skR, "odoh query")
  return context
```

decrypt_query_body: Decrypt an Oblivious DoH query.

```
def decrypt_query_body(context, key_id, Q_encrypted):
  aad = 0x01 || len(key_id) || key_id
  enc || ct = Q_encrypted
  Q_plain, error = context.Open(aad, ct)
  return Q_plain, error
```

derive_secrets: Derive keying material used for encrypting an Oblivious DoH response.

```
def derive_secrets(context, Q_plain, response_nonce):
  secret = context.Export("odoh response", Nk)
  salt = Q_plain || len(response_nonce) || response_nonce
  prk = Extract(salt, secret)
  key = Expand(odoh_prk, "odoh key", Nk)
  nonce = Expand(odoh_prk, "odoh nonce", Nn)
  return key, nonce
```

The "random(N)" function returns "N" cryptographically secure random bytes from a good source of entropy [RFC4086]. The "max(A, B)" function returns "A" if "A > B", and "B" otherwise.

encrypt_response_body: Encrypt an Oblivious DoH response.

```
def encrypt_response_body(R_plain, aead_key, aead_nonce, response_nonce):
  aad = 0x02 || len(response_nonce) || response_nonce
  R_encrypted = Seal(aead_key, aead_nonce, aad, R_plain)
  return R_encrypted
```

8. Oblivious Client Behavior

Let "M" be a DNS message (query) a client wishes to protect with Oblivious DoH. When sending an Oblivious DoH Query for resolving "M" to an Oblivious Target with "ObliviousDoHConfigContents" "config", a client does the following:

1. Create an "ObliviousDoHQuery" structure, carrying the message M and padding, to produce Q_plain.

2. Deserialize "config.public_key" to produce a public key pkR of type "config.kem_id".

3. Compute the encrypted message as "Q_encrypted = encrypt_query_body(pkR, key_id, Q_plain)", where "key_id" is as computed in Section 7. Note also that "len(key_id)" outputs the length of "key_id" as a two-byte unsigned integer.

4. Output a ObliviousDoHMessage message "Q" where "Q.message_type = 0x01", "Q.key_id" carries "key_id", and "Q.encrypted_message = Q_encrypted".

The client then sends "Q" to the Proxy according to Section 4.1. Once the client receives a response "R", encrypted as specified in Section 9, it uses "decrypt_response_body" to decrypt "R.encrypted_message" (using "R.key_id" as a nonce) and produce R_plain. Clients MUST validate "R_plain.padding" (as all zeros) before using "R_plain.dns_message".

9. Oblivious Target Behavior

Targets that receive a Query message Q decrypt and process it as follows:

1. Look up the "ObliviousDoHConfigContents" according to "Q.key_id". If no such key exists, the Target MAY discard the query, and if so, it MUST return a 400 (Client Error) response to the Proxy. Otherwise, let "skR" be the private key corresponding to this public key, or one chosen for trial decryption.

2. Compute "context = setup_query_context(skR, Q.key_id, Q.encrypted_message)".

3. Compute "Q_plain, error = decrypt_query_body(context, Q.key_id, Q.encrypted_message)".

4. If no error was returned, and "Q_plain.padding" is valid (all zeros), resolve "Q_plain.dns_message" as needed, yielding a DNS message M. Otherwise, if an error was returned or the padding was invalid, return a 400 (Client Error) response to the Proxy.

5. Create an "ObliviousDoHResponseBody" structure, carrying the message "M" and padding, to produce "R_plain".

6. Create a fresh nonce "response_nonce = random(max(Nn, Nk))".

7. Compute "aead_key, aead_nonce = derive_secrets(context, Q_plain, response_nonce)".

8. Compute "R_encrypted = encrypt_response_body(R_plain, aead_key, aead_nonce, response_nonce)". The "key_id" field used for encryption carries "response_nonce" in order for clients to derive the same secrets. Also, the "Seal" function is that which is associated with the HPKE AEAD.

9. Output a "ObliviousDoHMessage" message "R" where "R.message_type = 0x02", "R.key_id = response_nonce", and "R.encrypted_message = R_encrypted".

The Target then sends "R" in a 2xx (Successful) response to the Proxy; see Section 4.3. The Proxy forwards the message "R" without modification back to the client as the HTTP response to the client's original HTTP request. In the event of an error (non 2xx status code), the Proxy forwards the Target error to the client; see Section 4.3.

10. Compliance Requirements

Oblivious DoH uses draft-08 of HPKE for public key encryption [I-D.irtf-cfrg-hpke]. In the absence of an application profile standard specifying otherwise, a compliant Oblivious DoH implementation MUST support the following HPKE cipher suite:

* KEM: DHKEM(X25519, HKDF-SHA256) (see [I-D.irtf-cfrg-hpke], Section 7.1)

* KDF: HKDF-SHA256 (see [I-D.irtf-cfrg-hpke], Section 7.2)

* AEAD: AES-128-GCM (see [I-D.irtf-cfrg-hpke], Section 7.3)

11. Security Considerations

DISCLAIMER: this is a work in progress draft and has not yet seen significant security analysis.

Oblivious DoH aims to keep knowledge of the true query origin and its contents known to only clients. As a simplified model, consider a case where there exists two clients C1 and C2, one proxy P, and one target T. Oblivious DoH assumes an extended Dolev-Yao style attacker which can observe all network activity and can adaptively compromise either P or T, but not C1 or C2. Once compromised, the attacker has access to all session information and private key material. (This generalizes to arbitrarily many clients, proxies, and targets, with the constraints that not all targets and proxies are simultaneously compromised, and at least two clients are left uncompromised.) The attacker is prohibited from sending client identifying information, such as IP addresses, to targets. (This would allow the attacker to trivially link a query to the corresponding client.)

In this model, both C1 and C2 send an Oblivious DoH queries Q1 and Q2, respectively, through P to T, and T provides answers A1 and A2. The attacker aims to link C1 to (Q1, A1) and C2 to (Q2, A2), respectively. The attacker succeeds if this linkability is possible without any additional interaction. (For example, if T is compromised, it may return a DNS answer corresponding to an entity it controls, and then observe the subsequent connection from a client, learning its identity in the process. Such attacks are out of scope for this model.)

Oblivious DoH security prevents such linkability. Informally, this means:

1. Queries and answers are known only to clients and targets in possession of the corresponding response key and HPKE keying material. In particular, proxies know the origin and destination of an oblivious query, yet do not know the plaintext query. Likewise, targets know only the oblivious query origin, i.e., the proxy, and the plaintext query. Only the client knows both the plaintext query contents and destination.

2. Target resolvers cannot link queries from the same client in the absence of unique per-client keys.

Traffic analysis mitigations are outside the scope of this document. In particular, this document does not recommend padding lengths for ObliviousDoHQuery and ObliviousDoHResponse messages. Implementations SHOULD follow the guidance for choosing padding length in [RFC8467].

Oblivious DoH security does not depend on proxy and target indistinguishability. Specifically, an on-path attacker could determine whether a connection a specific endpoint is used for oblivious or direct DoH queries. However, this has no effect on confidentiality goals listed above.

11.1. Denial of Service

Malicious clients (or proxies) may send bogus Oblivious DoH queries to targets as a Denial-of-Service (DoS) attack. Target servers may throttle processing requests if such an event occurs. Additionally, since Targets provide explicit errors upon decryption failure, i.e., if ciphertext decryption fails or if the plaintext DNS message is malformed, Proxies may throttle specific clients in response to these errors.

Malicious Targets or Proxies may send bogus answers in response to Oblivious DoH queries. Response decryption failure is a signal that either the proxy or target is misbehaving. Clients can choose to stop using one or both of these servers in the event of such failure. However, as above, malicious Targets and Proxies are out of scope for the threat model.

11.2. Proxy Policies

Proxies are free to enforce any forwarding policy they desire for clients. For example, they may only forward requests to known or otherwise trusted targets. Proxies that do not have an allow list of targets can attempt to determine if the specified target is a valid Oblivious DoH target by querying for the target configuration as specified in Section 5.

11.3. General Proxy Services

Using DoH over anonymizing proxy services such as Tor would also achieve the desired goal of separating query origins from their contents. However, there are several reasons why such systems are undesirable in comparison Oblivious DoH:

1. Tor is also meant as a generic connection-level anonymity system, and thus seems overly complex and costly for the purpose of proxying individual DoH queries. In contrast, Oblivious DoH is a lightweight extension to standard DoH, implemented as an application-layer proxy, that can be enabled as a default mode for users which need increased privacy.

2. As a one-hop proxy, Oblivious DoH encourages connection-less proxies to mitigate client query correlation with few round-trips. In contrast, multi-hop systems such as Tor often run secure connections (TLS) end-to-end, which means that DoH servers could track queries over the same connection. Using a fresh DoH connection per query would incur a non-negligible penalty in connection setup time.

12. IANA Considerations

12.1. Oblivious DoH Message Media Type

This document registers a new media type, "application/oblivious-dns-message".

Type name: application

Subtype name: oblivious-dns-message

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: This is a binary format, containing encrypted DNS requests and responses, as defined in this document.

Security considerations: See this document. The content is an encrypted DNS message, and not executable code.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document.

Applications that use this media type: This media type is intended to be used by clients wishing to hide their DNS queries when using DNS over HTTPS.

Additional information: None

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: None

Author: IETF

Change controller: IETF

12.2. Oblivious DoH Public Key DNS Parameter

This document adds a parameter ("odohconfig") to the "Service Binding (SVCB) Parameter" registry [I-D.ietf-dnsop-svcb-https]. The allocation request is 32769, taken from the to the First Come First Served range.

If present, the "odohconfig" parameter contains a ObliviousDoHConfigs structure. In wire format, the value of the parameter is an ObliviousDoHConfigs vector, including the redundant length prefix. In presentation format, the value is encoded in [base64].

Name: odohconfig

SvcParamKey: 32769

Meaning: An ObliviousDoHConfigs structure.

Reference: This document.

13. Acknowledgments

This work is inspired by Oblivious DNS [I-D.annee-dprive-oblivious-dns]. Thanks to all of the authors of that document. Thanks to Nafeez Ahamed, Elliot Briggs, Marwan Fayed, Frederic Jacobs, Tommy Jensen Paul Schmitt, Brian Swander, Tanya Verma, and Peter Wu for the feedback and input.

14. References

14.1. Normative References

[base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[I-D.ietf-dnsop-svcb-https]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-02, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-svcb-https-02.txt>.

[I-D.ietf-httpbis-proxy-status]
Nottingham, M. and P. Sikora, "The Proxy-Status HTTP Response Header Field", Work in Progress, Internet-Draft, draft-ietf-httpbis-proxy-status-02, 11 August 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-proxy-status-02.txt>.

[I-D.irtf-cfrg-hpke]
Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-hpke-07, 16 December 2020, <http://www.ietf.org/internet-drafts/draft-irtf-cfrg-hpke-07.txt>.

[I-D.pauly-add-resolver-discovery]
Pauly, T., Kinnear, E., Wood, C., McManus, P., and T. Jensen, "Adaptive DNS Resolver Discovery", Work in Progress, Internet-Draft, draft-pauly-add-resolver-discovery-01, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-pauly-add-resolver-discovery-01.txt>.

[I-D.pauly-dprive-adaptive-dns-privacy]
Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Adaptive DNS: Improving Privacy of Name Resolution", Work in Progress, Internet-Draft, draft-pauly-dprive-adaptive-dns-privacy-01, 1 November 2019, <http://www.ietf.org/internet-drafts/draft-pauly-dprive-adaptive-dns-privacy-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[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, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <https://www.rfc-editor.org/info/rfc8467>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

14.2. Informative References

[I-D.annee-dprive-oblivious-dns] Edmundson, A., Schmitt, P., Feamster, N., and A. Mankin, "Oblivious DNS - Strong Privacy for DNS Queries", Work in Progress, Internet-Draft, draft-annee-dprive-oblivious-dns-00, 2 July 2018, <http://www.ietf.org/internet-drafts/draft-annee-dprive-oblivious-dns-00.txt>.

[RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", RFC 7239, DOI 10.17487/RFC7239, June 2014, <https://www.rfc-editor.org/info/rfc7239>.

Authors' Addresses

Eric Kinnear
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: ekinnear@apple.com

Patrick McManus
Fastly

Email: mcmanus@ducksong.com

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

Tanya Verma
Cloudflare
101 Townsend St
San Francisco,
United States of America

Email: vermatanyax@gmail.com

Christopher A. Wood
Cloudflare
101 Townsend St
San Francisco,
United States of America

Email: caw@heapingbits.net