Workgroup:	dprive
Internet-Draft:	draft-ietf-dprive-unilateral-probing-latest
Published:	20 February 2024
Intended Status:	Experimental
Expires:	23 August 2024

Authors: D. K. Gillmor, Ed. *ACLU* J. Salazar, Ed. P. Hoffman, Ed. *ICANN*

Unilateral Opportunistic Deployment of Encrypted Recursive-to-Authoritative DNS

Abstract

This document sets out steps that DNS servers (recursive resolvers and authoritative servers) can take unilaterally (without any coordination with other peers) to defend DNS query privacy against a passive network monitor. The steps in this document can be defeated by an active attacker, but should be simpler and less risky to deploy than more powerful defenses.

The goal of this document is to simplify and speed deployment of opportunistic encrypted transport in the recursive-to-authoritative hop of the DNS ecosystem. Wider easy deployment of the underlying encrypted transport on an opportunistic basis may facilitate the future specification of stronger cryptographic protections against more powerful attacks.

The RFC Editor will remove this note

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://dkg.gitlab.io/dprive-unilateral-probing/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-dprive-unilateral-probing/.

Discussion of this document takes place on the DPRIVE Working Group mailing list (mailto:dns-privacy@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/dns-privacy/. Subscribe at https://www.ietf.org/mailman/listinfo/dns-privacy/.

Source for this draft and an issue tracker can be found at https://gitlab.com/dkg/dprive-unilateral-probing.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document aims to provide guidance to DNS implementers and operators who want to simply enable protection against passive network observers.

In particular, it focuses on mechanisms that can be adopted unilaterally by recursive resolvers and authoritative servers, without any explicit coordination with the other parties. This guidance provides opportunistic security (see [RFC7435]) -- encrypting things that would otherwise be in the clear, without interfering with or weakening stronger forms of security.

The document also briefly introduces (but does not try to specify) how a future protocol might permit defense against an active attacker in Appendix C.

The protocol described here offers three concrete advantages to the DNS ecosystem:

- Protection from passive attackers of DNS queries in transit between recursive and authoritative servers.
- A roadmap for gaining real-world experience at scale with encrypted protections of this traffic.
- A bridge to some possible future protection against a more powerful attacker.

1.1. Requirements Language

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.

1.2. Terminology

Unilateral: capable of opportunistic probing without external coordination with any of the other parties

Do53: traditional cleartext DNS over port 53 ([RFC1035])

DoQ: DNS-over-QUIC ([RFC9250])

DoT: DNS-over-TLS ([RFC7858])

Encrypted transports: DoQ and DoT collectively

2. Priorities

This document aims to mitigate potential impacts of the described protocol and to aid implementors in selecting between possible DNS protocol choices.

2.1. Minimizing Negative Impacts

The protocol described in this document aims to minimize potentially negative impacts caused by the probing of encrypted transports for the systems that adopt these guidelines, for the parties that they communicate with, and for uninvolved third parties. The negative impacts that this protocol specifically tries to minimize are:

- excessive bandwidth use
- excessive use of computational resources (CPU and memory in particular)
- the potential for amplification attacks (where DNS resolution infrastructure is wielded as part of a DoS attack)

2.2. Protocol Choices

Although this document focuses specifically on strategies used by DNS servers, it does not go into detail on the specific protocols used because those protocols, in particular DoT and DoQ, are described in other documents. The DoT specification ([RFC7858]) says that it:

> focuses on securing stub-to-recursive traffic, as per the charter of the DPRIVE Working Group. It does not prevent future applications of the protocol to recursive-to-authoritative traffic.

It further says:

> It might work equally between recursive clients and authoritative servers.

The DoQ specification ([RFC9250]) says:

> For the recursive to authoritative scenario, authentication requirements are unspecified at the time of writing and are the subject of ongoing work in the DPRIVE WG.

The protocol described in this document specifies the use of DoT and DoQ without authentication of the server.

This document does not pursue the use of DNS-over-HTTPS, commonly called DoH ([RFC8484]), in this context because a DoH client needs to know the path part of a DoH endpoint URL, and there are currently no mechanisms for a DNS recursive resolver to predict the path on its own, in an opportunistic or unilateral fashion, without incurring an excessive use of resources. If such mechanisms are later defined, the protocol in this document can be updated to accommodate them.

3. Guidance for Authoritative Servers

The protocol described in this document is **OPTIONAL** for authoritative servers. An authoritative server choosing to implement the protocol described in this document **MUST** implement at least one of DoT or DoQ on port 853.

An authoritative server choosing to implement the protocol described in this document **MAY** require clients to use ALPN (Application-Layer Protocol Negotiation, [RFC7301]). The ALPN strings the client will use are given in Section 4.4.

An authoritative server implementing DoT or DoQ **MUST** populate the response from the same authoritative zone data as the unencryped DNS transports. Encrypted transports have their own characteristic response size that might be different from the unencrypted DNS transports, so response sizes and related options (e.g., EDNS0) and flags (e.g., TC bit) might vary based on the transport. In other words, the content of the responses to a particular query **MUST** be the same regardless of the type of transport.

3.1. Pooled Authoritative Servers Behind a Load Balancer

Some authoritative DNS servers are structured as a pool of authoritatives standing behind a load-balancer that runs on a single IP address, forwarding queries to members of the pool. In such a deployment, individual members of the pool typically get updated independently from each other.

A recursive resolver following the guidance in Section 4 and interacting with such a pool likely does not know that it is a pool. If some members of the pool follow the protocol specified in this document while others do not, the recursive client might see the pool as a single authoritative server that sometimes offers and sometimes refuses encrypted transport.

To avoid incurring additional minor timeouts for such a recursive resolver, the pool operator **SHOULD**:

- ensure that all members of the pool enable the same encrypted transport(s) within the span of a few seconds (such as within 30 seconds), or
- ensure that the load balancer maps client requests to pool members based on client IP addresses, or
- use a load-balancer that forwards queries/connections on encrypted transports to only those members of the pool known (e.g., via monitoring) to support the given encrypted transport.

Similar concerns apply to authoritative servers responding from an anycast IP address. As long as the pool of servers is in a heterogeneous state, any flapping route that switches a given client IP address to a different responder risks incurring an additional timeout. Frequent changes of routing for anycast listening IP addresses are also likely to cause problems for TLS, TCP, or QUIC connection state as well, so stable routes are important to ensure that the service remains available and responsive. The servers in a pool can share session information to increase the likelihood of successful resumptions.

3.2. Authentication

For unilateral deployment, an authoritative server does not need to offer any particular form of authentication.

One simple deployment approach would just be to provide a self-issued, regularly-updated X.509 certificate. Whether the certificates used are short-lived or long-lived is up to the deployment. This mechanism is supported by many TLS and QUIC clients, and will be acceptable for any opportunistic connection. The server could provide a normal PKI-based certificate, but there is no advantage to doing so at this time.

3.3. Server Name Indication

An authoritative DNS server that wants to handle unilateral queries **MAY** rely on Server Name Indication (SNI) to select alternate server credentials. However, such a server **MUST NOT** serve resource records that differ based on SNI (or on the lack of SNI) provided by the client, because a probing recursive resolver that offers SNI might or might not have used the right server name to get the records it is looking for.

3.4. Resource Exhaustion

A well-behaved recursive resolver may keep an encrypted connection open to an authoritative server, to amortize the costs of connection setup for both parties.

However, some authoritative servers may have insufficient resources available to keep many connections open concurrently.

To keep resources under control, authoritative servers should proactively manage their encrypted connections. Section 5.5 of [RFC9250] ("Connection Handling") offers useful guidance for servers managing DoQ connections. Section 3.4 of [RFC7858] offers useful guidance for servers managing DoT connections.

An authoritative server facing unforeseen resource exhaustion **SHOULD** cleanly close open connections from recursive resolvers based on the authoritative's preferred prioritization.

In the case of unanticipated resource exhaustion, close connections until resources are back in control. A reasonable prioritization scheme would be to close connections with no outstanding queries, ordered by idle time (longest idle time gets closed first), then close connections with outstanding queries, ordered by age of outstanding query (oldest outstanding query gets closed first).

When resources are especially tight, the authoritative server may also decline to accept new connections over encrypted transport.

3.5. Pad Responses to Mitigate Traffic Analysis

To increase the anonymity set for each response, the authoritative server **SHOULD** use a sensible padding mechanism for all responses it sends when possible. The ability for the authoritative server to add padding might be limited, such as by not receiving an EDNS(0) option in the query. Specifically, a DoT server **SHOULD** use EDNS(0) padding [RFC7830] if possible, and a DoQ server **SHOULD** follow the guidance in Section 5.4 of [RFC9250]. How much to pad is out of scope of this document, but a reasonable suggestion can be found in [RFC8467].

4. Guidance for Recursive Resolvers

The protocol described in this document is **OPTIONAL** for recursive resolvers. This section outlines a probing policy suitable for unilateral adoption by any recursive resolver. Following this policy should not result in failed resolutions or significant delays.

4.1. High-level Overview

In addition to querying on Do53, the recursive resolver will try either or both of DoT and DoQ concurrently. The recursive resolver remembers what opportunistic encrypted transport protocols have worked recently based on a (clientIP, serverIP, protocol) tuple.

If a query needs to go to a given authoritative server, and the recursive resolver remembers a recent successful encrypted transport to that server, then it doesn't send the query over Do53 at all. Rather, it only sends the query using the encrypted transport protocol that was recently shown to be good.

If the encrypted transport protocol fails, the recursive resolver falls back to Do53 for that serverIP. When any encrypted transport fails, the recursive resolver remembers that failure for a reasonable amount of time to avoid flooding a non-compatible server with requests that it cannot accept. The description of how an encrypted transport protocol fails is in Section 4.6.4 and the sections following that.

See the subsections below for a more detailed description of this protocol.

4.2. Maintaining Authoritative State by IP Address

In designing a probing strategy, the recursive resolver could record its knowledge about any given authoritative server with different strategies, including at least:

- the authoritative server's IP address,
- the authoritative server's name (the NS record used), or
- the zone that contains the record being looked up.

This document encourages the first strategy, to minimize timeouts or accidental delays, and does not describe the other two strategies.

A timeout (accidental delay) is most likely to happen when the recursive client believes that the authoritative server offers encrypted transport, but the actual server reached declines encrypted transport (or worse, filters the incoming traffic and does not even respond with an ICMP destination port unreachable message, such as during rate limiting).

By associating the state with the authoritative IP address, the client can minimize the number of accidental delays introduced (see also Section 3.1 and Section 4.5).

For example, consider an authoritative server named `ns0.example.com` that is served by two installations, one at `2001:db8::7` that follows this guidance, and one at `2001:db8::8` that is a legacy (cleartext port 53-only) deployment. A recursive client who associates state with the `NS` name and reaches `2001:db8::7` first will "learn" that `ns0.example.com` supports encrypted transport. A subsequent query over encrypted transport dispatched to `2001:db8::8` would fail, potentially delaying the response.

4.3. Overall Recursive Resolver Settings

A recursive resolver implementing the protocol in this document needs to set system-wide values for some default parameters. These parameters may be set independently for each supported encrypted transport, though a simple implementation may keep the parameters constant across encrypted transports.

Name	Description	Suggested Default
persistence	How long should the recursive resolver remember successful encrypted transport connections?	3 days (259200 seconds)
damping	How long should the recursive resolver remember unsuccessful encrypted transport connections?	1 day (86400 seconds)
timeout	How long should the recursive resolver wait for an initiated encrypted connection to complete?	4 seconds

Table 1: Recursive resolver system parameters per encrypted transport

This document uses the notation `<transport>-foo` to refer to the `foo` parameter for the encrypted transport `<transport>`. For example, `DoT-persistence` would indicate the length of time that the recursive resolver will remember that an authoritative server had a successful connection over `DoT`. Additionally, when describing an arbitrary encrypted transport, we use `E` in place of `<transport>` to generically mean whatever encrypted transport is in use. For example, when handling a query sent over encrypted transport `E`, a reference to `E-timeout` should be understood to mean `DoT-timeout` if the query is sent over `DoT`, and to mean `DoQ-timeout` if the query is sent over `DoQ`.

This document also assumes that the recursive resolver maintains a list of outstanding cleartext queries destined for the authoritative server's IP address `X`. This list is referred to as `Do53-queries[X]`. This document does not attempt to describe the specific operation of sending and receiving cleartext DNS queries (Do53) for a recursive resolver. Instead it describes a "bolt-on" mechanism that extends the recursive resolver's operation on a few simple hooks into the recursive resolver's existing handling of Do53.

Implementers or deployers of DNS recursive resolvers that follow the strategies in this document are encouraged to publish their preferred values of these parameters.

4.4. Recursive Resolver Requirements

To follow this guidance, a recursive resolver **MUST** implement at least one of either DoT or DoQ in its capacity as a client of authoritative nameservers. A recursive resolver **SHOULD** implement the client side of DoT. A recursive resolver **SHOULD** implement the client side of DoQ.

DoT queries from the recursive resolver **MUST** target TCP port 853, using an ALPN of "`dot`". DoQ queries from the recursive resolver **MUST** target UDP port 853, using an ALPN of "`doq`".

While this document focuses on the recursive-to-authoritative hop, a recursive resolver implementing these strategies **SHOULD** also accept queries from its clients over some encrypted transport unless it only accepts queries from localhost.

4.5. Authoritative Server Encrypted Transport Connection State

The recursive resolver **SHOULD** keep a record of the state for each authoritative server it contacts, indexed by the IP address of the authoritative server and the encrypted transports supported by the recursive resolver.

Note that the recursive resolver might record this per-authoritative-IP state for each source IP address it uses as it sends its queries. For example, if a recursive resolver can send a packet to authoritative servers from IP addresses `2001:db8::100` and `2001:db8::200`, it could keep two distinct sets of per-authoritative-IP state, one for each source address it uses, if the recursive resolver knows the addresses in use. Keeping these state tables distinct for each source address makes it possible for a pooled authoritative server behind a load balancer to do a partial rollout while minimizing accidental timeouts (see Section 3.1).

In addition to tracking the state of connection attempts and outcomes, a recursive resolver **SHOULD** record the state of established sessions for encrypted protocols. The details of how sessions are identified is dependent on the transport protocol implementation (such as TLS session ticket or TLS session ID, QUIC connection ID, and so on). The use of session resumption as recommended here is limited somewhat because the tickets are only stored within the context defined by the (clientIP, serverIP, protocols) tuples used to track client-server interaction by the recursive resolver in a table like the one below. However, session resumption still offers the ability to optimize the handshake in some circumstances.

Each record should contain the following fields for each supported encrypted transport, each of which would initially be `null`:

Name	Description	Retain Across Restart
session	The associated state of any existing, established session (the structure of this value is dependent on the encrypted transport implementation). If session is not null, it may be in one of two states: pending or established	no
initiated	Timestamp of most recent connection attempt	yes
completed	Timestamp of most recent completed handshake (which can include one where an existing session is resumed)	yes
status	Enumerated value of success or fail or timeout, associated with the completed handshake	yes
last-response	A timestamp of the most recent response received on the connection	yes
resumptions	A stack of resumption tickets (and associated parameters) that could be used to resume a prior successful session	yes
queries	A queue of queries intended for this authoritative server, each of which has additional status early, unsent, or sent	no
last-activity	A timestamp of the most recent activity on the connection	no

Table 2: Recursive resolver state per authoritative IP, per encrypted transport

Note that the session fields in aggregate constitute a pool of open connections to different servers.

With the exception of the session, queries, and last-activity fields, this cache information should be kept across restart of the server unless explicitly cleared by administrative action.

This document uses the notation E-foo[X] to indicate the value of field foo for encrypted transport E to IP address X.

For example, DoT-initiated[192.0.2.4] represents the timestamp when the most recent DoT connection packet was sent to IP address 192.0.2.4.

This document uses the notation any-E-queries to indicate any query on an encrypted transport.

4.6. Probing Policy

When a recursive resolver discovers the need for an authoritative lookup to an authoritative DNS server using that servers's IP address `X`, it retrieves the connection state records described in Section 4.5 associated with `X` from its cache.

The subsections that follow offer pseudocode that corresponds roughly to an asynchronous programming model for a recursive resolver's interactions with authoritative servers. The following subsections also presume that the time of the discovery of the need for lookup is time `T0`.

If any of the records discussed here are absent, they are treated as `null`.

The recursive resolver must decide whether to initially send a query over Do53, or over any of the supported encrypted transports (DoT or DoQ).

Note that a recursive resolver might initiate this query via any or all of the known transports. When multiple queries are sent, the initial packets for each connection can be sent concurrently, similar to "Happy Eyeballs" ([RFC8305]). However, unlike Happy Eyeballs, when one transport succeeds, the other connections do not need to be terminated, but can instead be continued to establish whether the IP address `X` is capable of communicating on the relevant transport.

4.6.1. Sending a Query over Do53

For any of the supported encrypted transports `E`, if either of the following holds true, the recursive resolver **SHOULD NOT** send a query to `X` over Do53:

- `E-session[X]` is in the `established` state, or
- `E-status[X]` is `success`, and `(T0 - E-last-response[X]) < persistence`

This indicates that one successful connection to a server that the client then closed cleanly would result in the client not sending the next query over Do53.

Otherwise, if there is no outstanding session for any encrypted transport, and the last successful encrypted transport connection was long ago, the recursive resolver sends a query to `X` over Do53. When it does so, it inserts a handle for the query in `Do53-queries[X]`.

4.6.2. Receiving a Response over Do53

When any response `R` (a well-formed DNS response, asynchronous timeout, asynchronous destination port unreachable, etc) for query `Q` arrives at the recursive resolver in cleartext sent over Do53 from authoritative server with IP address `X`, the recursive resolver should:

If Q is not in Do53-queries[X]:

- Process R no further (do not respond to a cleartext response to a query that is not outstanding)

Otherwise, if Q was marked as already processed:

- Remove Q from Do53-queries[X]
- Discard any content from the response and process R no further

If R is a well-formed DNS response:

- Remove Q from Do53-queries[X]
- R is further processed by the recursive resolver
- For each supported encrypted transport E:
    - If Q is in E-queries[X]:
        - Mark Q as already processed

But if R is malformed, or a failure (e.g., a timeout or destination port unreachable):

- if Q is not in any of any-E-queries[X]:
    - Treat this as a failed query (i.e., follow the resolver's policy for unresponsive or non-compliant authoritatives, such as falling back to another authoritative server, returning SERVFAIL to the requesting client, and so on)

4.6.3. Initiating a Connection over Encrypted Transport

If any E-session[X] is in the established state, the recursive resolver **SHOULD NOT** initiate a new or resume a previous connection to X over Do53 or E, but should instead send queries to X through the existing session (see Section 4.6.8).

If the recursive resolver prefers one encrypted transport over another, but only the unpreferred encrypted transport is in the established state, it **MAY** also initiate a new connection to X over its preferred encrypted transport while concurrently sending the query over the established encrypted transport E.

Before considering whether to initiate a new connection over an encrypted transport, the timer should be examined, and its state possibly refreshed, for encrypted transport E to authoritative IP address X:

- if E-session[X] is in state pending, and

- T_0 - E-initiated[X] > E-timeout, then
  - set E-session[X] to null and
  - set E-status[X] to timeout

When resources are available to attempt a new encrypted transport, the recursive resolver should only initiate a new connection to X over E as long as one of the following holds true:

- E-status[X] is success, or
- E-status[X] is either fail or timeout, and (T_0 - E-completed[X]) > damping, or
- E-status[X] is null and E-initiated[X] is null

When initiating a session to X over encrypted transport E, if E-resumptions[X] is not empty, one ticket should be popped off the stack and used to try to resume a previous session. Otherwise, the initial Client Hello handshake should not try to resume any session.

When initiating a connection, the recursive resolver should take the following steps:

- set E-initiated[X] to T_0
- store a handle for the new session (which should have pending state) in E-session[X]
- insert a handle for the query that prompted this connection in E-queries[X], with status unsent or early, as appropriate (see below).

4.6.3.1. Early Data

Modern encrypted transports like TLS 1.3 offer the chance to send "early data" from the client in the initial Client Hello in some contexts. A recursive resolver that initiates a connection over an encrypted transport according to this guidance in a context where early data is possible **SHOULD** send the DNS query that prompted the connection in the early data, according to the sending guidance in Section 4.6.8.

If it does so, the status of Q in E-queries[X] should be set to early instead of unsent.

4.6.3.2. Resumption Tickets

When initiating a new connection (whether by resuming an old session or not), the recursive resolver **SHOULD** request a session resumption ticket from the authoritative server. If the authoritative server supplies a resumption ticket, the recursive resolver pushes it into the stack at E-resumptions[X].

4.6.3.3. Server Name Indication

For modern encrypted transports like TLS 1.3, most client implementations expect to send a Server Name Indication (SNI) in the Client Hello.

There are two complications with selecting or sending SNI in this unilateral probing:

- Some authoritative servers are known by more than one name; selecting a single name to use for a given connection may be difficult or impossible.
- In most configurations, the contents of the SNI field is exposed on the wire to a passive adversary. This potentially reveals additional information about which query is being made, based on the NS of the query itself.

To avoid additional leakage and complexity, a recursive resolver following this guidance **SHOULD NOT** send SNI to the authoritative when attempting encrypted transport.

If the recursive resolver needs to send SNI to the authoritative for some reason not found in this document, using Encrypted Client Hello ([I-D.ietf-tls-esni]) would reduce leakage.

4.6.3.4. Authoritative Server Authentication

Because this probing policy is unilateral and opportunistic, the client connecting under this policy **MUST** accept any certificate presented by the server. If the client cannot verify the server's identity, it **MAY** use that information for reporting, logging, or other analysis purposes. But it **MUST NOT** reject the connection due to the authentication failure, as the result would be falling back to cleartext, which would leak the content of the session to a passive network monitor.

4.6.4. Establishing an Encrypted Transport Connection

When an encrypted transport connection actually completes (e.g., the TLS handshake completes) at time T_1, the recursive resolver sets E-completed[X] to T_1 and does the following:

If the handshake completed successfully:

- update E-session[X] so that it is in state established
- set E-status[X] to success
- set E-last-response[X] to T_1
- set E-completed[X] to T_1
- for each query Q in E-queries[X]:
  - if early data was accepted and Q is early,
    - set the status of Q to sent
  - otherwise:
    - send Q through the session (see Section 4.6.8), and set the status of Q to sent

4.6.5. Failing to Establish an Encrypted Transport Connection

If, at time T_2 an encrypted transport handshake completes with a failure (e.g., a TLS alert),

- set E-session[X] to null

- set E-status[X] to fail
- set E-completed[X] to T_2
- for each query Q in E-queries[X]:
  - if Q is not present in any other any-E-queries[X] or in Do_{53}-queries[X], add Q to Do_{53}-queries[X] and send query Q to X over Do53.

Note that this failure will trigger the recursive resolver to fall back to cleartext queries to the authoritative server at IP address X. It will retry encrypted transport to X once the damping timer has elapsed.

4.6.6. Encrypted Transport Failure

Once established, an encrypted transport might fail for a number of reasons (e.g., decryption failure, or improper protocol sequence).

If this happens:

- set E-session[X] to null
- set E-status[X] to fail
- for each query Q in E-queries[X]:
  - if Q is not present in any other any-E-queries[X] or in Do_{53}-queries[X], add Q to Do_{53}-queries[X] and send query Q to X over Do53.

Note that this failure will trigger the recursive resolver to fall back to cleartext queries to the authoritative server at IP address X. It will retry encrypted transport to X once the damping timer has elapsed.

4.6.7. Handling Clean Shutdown of an Encrypted Transport Connection

At time T_3, the recursive resolver may find that authoritative server X cleanly closes an existing outstanding connection (most likely due to resource exhaustion, see Section 3.4).

When this happens:

- set E-session[X] to null
- for each query Q in E-queries[X]:
  - if Q is not present in any other any-E-queries[X] or in Do_{53}-queries[X], add Q to Do_{53}-queries[X] and send query Q to X over Do53.

Note that this premature shutdown will trigger the recursive resolver to fall back to cleartext queries to the authoritative server at IP address X. Any subsequent query to X will retry the encrypted connection promptly.

4.6.8. Sending a Query over Encrypted Transport

When sending a query to an authoritative server over encrypted transport at time T_4, the recursive resolver should take a few reasonable steps to ensure privacy and efficiency.

After sending query Q, the recursive resolver should ensure that Q's state in E-queries[X] is set to sent.

The recursive resolver also sets E-last-activity[X] to T_4.

In addition, the recursive resolver should consider the guidance in the following sections.

4.6.8.1. Pad Queries to Mitigate Traffic Analysis

To increase the anonymity set for each query, the recursive resolver **SHOULD** use a sensible padding mechanism for all queries it sends. Specifically, a DoT client **SHOULD** use EDNS(0) padding [RFC7830], and a DoQ client **SHOULD** follow the guidance in Section 5.4 of [RFC9250]. How much to pad is out of scope of this document, but a reasonable suggestion can be found in [RFC8467].

4.6.8.2. Send Queries in Separate Channels

When multiple queries are multiplexed on a single encrypted transport to a single authoritative server, the recursive resolver **SHOULD** pipeline queries and **MUST** be capable of receiving responses out of order. For guidance on how to best achieve this on a given encrypted transport, see Section 6.2.1.1 of [RFC7766] (for DoT) and Section 5.6 of [RFC9250] (for DoQ).

4.6.9. Receiving a Response over Encrypted Transport

Even though session-level events on encrypted transports like clean shutdown (see Section 4.6.7) or encrypted transport failure (see Section 4.6.6) can happen, some events happen on encrypted transport that are specific to a query, not session-wide. This subsection describes how the recursive resolver deals with events related to a specific query.

When a query-specific response R (a well-formed DNS response or an asynchronous timeout) associated with query Q arrives at the recursive resolver over encrypted transport E from authoritative server with IP address X at time T_5, the recursive resolver should:

If Q is not in E-queries[X]:

- Discard the response and process R no further (do not respond to an encrypted response to a query that is not outstanding)

Otherwise:

- Remove Q from E-queries[X]

- Set E-last-activity[X] to T 5
- Set E-last-response[X] to T 5

If Q was marked as already processed:

- Discard the response and process the response no further

If R is a well-formed DNS response:

- R is further processed by the recursive resolver
- For each supported encrypted transport N other than E:
  - If Q is in N-queries[X]:
    - Mark Q as already processed
- If Q is in Do 5 3 -queries[X]:
  - Mark Q as already processed

But if R is malformed, or a failure (e.g., timeout):

- If Q is not in Do 5 3 -queries[X] or in any of any-E-queries[X]:
  - Treat this as a failed query (i.e., follow the resolver's policy for unresponsive or non-compliant authoritatives, such as falling back to another authoritative server, returning SERVFAIL to the requesting client, and so on)

4.6.10. Resource Exhaustion

To keep resources under control, a recursive resolver should proactively manage outstanding encrypted connections. Section 5.5 of [RFC9250] offers useful guidance for clients managing DoQ connections. Section 3.4 of [RFC7858] offers useful guidance for clients managing DoT connections.

Even with sensible connection management, a recursive resolver doing unilateral probing may find resources unexpectedly scarce, and may need to close some outstanding connections.

In such a situation, the recursive resolver **SHOULD** use a reasonable prioritization scheme to close outstanding connections.

One reasonable prioritization scheme would be:

- close outstanding established sessions based on E-last-activity[X] (oldest timestamp gets closed first)

Note that when resources are limited, a recursive resolver following this guidance may also choose not to initiate new connections for encrypted transport.

4.6.11. Maintaining Connections

Some recursive resolvers looking to amortize connection costs and to minimize latency **MAY** choose to synthesize queries to a particular authoritative server to keep an encrypted transport session active.

A recursive resolver that adopts this approach should try to align the synthesized queries with other optimizations. For example, a recursive resolver that "pre-fetches" a particular resource record to keep its cache "hot" can send that query over an established encrypted transport session.

4.6.12. Additional Tuning

A recursive resolver's state table for an authoritative server can contain additional information beyond what is described above. The recursive resolver might use that additional state to change the way it interacts with the authoritative server in the future. Some examples of additional state include:

- Whether the server accepts "early data" over a transport such as DoQ;
- Whether the server fails to respond to queries after the handshake succeeds;
- Tracking the round trip time of queries to the server;
- Tracking the number of timeouts (compared to the number of successful queries).

5. IANA Considerations

This document has no IANA considerations.

6. Privacy Considerations

6.1. Server Name Indication

A recursive resolver querying an authoritative server over DoT or DoQ that sends Server Name Indication (SNI) in the clear in the cryptographic handshake leaks information about the intended query to a passive network observer.

In particular, if two different zones refer to the same nameserver IP addresses via differently-named NS records, a passive network observer can distinguish the queries to one zone from the queries to the other.

Omitting SNI entirely, or using Encrypted Client Hello to hide the intended SNI, avoids this additional leakage. However, a series of queries that leak this information is still an improvement over cleartext.

6.2. Modelling the Probability of Encryption

Given that there are many parameter choices that can be made by recursive resolvers and authoritative servers, it is reasonable to consider the probability that queries would be encrypted. Such a measurement would also certainly be affected by the types of queries being sent by the recursive resolver, which in turn is also related to the types of queries that are sent to the recursive resolver by the stub resolvers and forwarders downstream. Doing this type of research would be valuable to the DNS community after initial implementation by a variety of recursive resolvers and authoritative servers because it would help assess the overall DNS privacy value of implementing the protocol. Thus, it would be useful if recursive resolvers and authoritative servers reported percentages of queries sent and received over the different transports.

7. Security Considerations

The guidance in this document provides defense against passive network monitors for most queries. It does not defend against active attackers. It can also leak some queries and their responses due to "happy eyeballs" optimizations when the recursive resolver's cache is cold.

Implementation of the guidance in this document should increase deployment of opportunistic encrypted DNS transport between recursive resolvers and authoritative servers at little operational risk.

However, implementers cannot rely on the guidance in this document for robust defense against active attackers, but should treat it as a stepping stone en route to stronger defense.

In particular, a recursive resolver following this guidance can easily be forced by an active attacker to fall back to cleartext DNS queries. Or, an active attacker could position itself as a machine-in-the-middle, which the recursive resolver would not defend against or detect due to lack of server authentication. Defending against these attacks without risking additional unexpected protocol failures would require signaling and coordination that are out of scope for this document.

This guidance is only one part of operating a privacy-preserving DNS ecosystem. A privacy-preserving recursive resolver should adopt other practices as well, such as QNAME minimization ([RFC9156]), local root zone ([RFC8806]), etc, to reduce the overall leakage of query information that could infringe on the client's privacy.

8. Operational Considerations

As recursive resolvers implement this protocol, authoritative servers will see more probing on port 853 of IP addresses that are associated with NS records. Such probing of an authoritative server should generally not cause any significant problems: if the authoritative server is not supporting this protocol, it will not respond on port 853, and if it is supporting this protocol, it will act accordingly.

However, a system that is a public recursive resolver that supports DoT and/or DoQ may also have an IP address that is associated with NS records. This could be accidental (such as a glue record with the wrong target address) or intentional. In such a case, a recursive resolver following this protocol will look for authoritative answers to ports 53 and 853 on that IP address, and DNS server answering on port 853 would need to be able to differentiate queries for recursive answers from queries for authoritative answers, for example by having the authoritative server handle all queries that have the Recursion Desired (RD) flag unset.

As discussed in Section 7, the protocol described in this document provides no defense against active attackers. On a network where a captive portal is operating, some communications may be actively intercepted, e.g., when the network tries to redirect a user to complete an interaction with a captive portal server. A recursive resolver operating on a node that performs captive portal detection and Internet connectivity checks **SHOULD** delay encrypted transport probes to authoritative servers until after the node's Internet connectivity check policy has been satisfied.

9. Acknowledgements

Many people contributed to the development of this document beyond the authors, including Alexander Mayrhofer, Brian Dickson, Christian Huitema, Dhruv Dhody, Eric Nygren, Erik Kline, Florian Obser, Haoyu Song, Jim Reid, Kris Shrishak, Peter Thomassen, Peter van Dijk, Ralf Weber, Rich Salz, Robert Evans, Sara Dickinson, Scott Hollenbeck, Stephane Bortzmeyer, Yorgos Thessalonikefs, and the DPRIVE working group.

10. References

10.1. Normative References

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

[RFC9250] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/rfc/rfc9250>.

[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/rfc/rfc7858>.

[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/rfc/rfc7301>.

10.2. Informative References

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

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

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

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

[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/rfc/rfc8467>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/rfc/rfc8305>.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-17, 9 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-17>.

[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/rfc/rfc7766>.

[RFC9156] Bortzmeyer, S., Dolmans, R., and P. Hoffman, "DNS Query Name Minimisation to Improve Privacy", RFC 9156, DOI 10.17487/RFC9156, November 2021, <https://www.rfc-editor.org/rfc/rfc9156>.

[RFC8806] Kumari, W. and P. Hoffman, "Running a Root Server Local to a Resolver", RFC 8806, DOI 10.17487/RFC8806, June 2020, <https://www.rfc-editor.org/rfc/rfc8806>.

[MTA-STS] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <https://www.rfc-editor.org/rfc/rfc8461>.

[DANE-SMTP] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <https://www.rfc-editor.org/rfc/rfc7672>.

[TLSRPT] Margolis, D., Brotman, A., Ramakrishnan, B., Jones, J., and M. Risher, "SMTP TLS Reporting", RFC 8460, DOI 10.17487/RFC8460, September 2018, <https://www.rfc-editor.org/rfc/rfc8460>.

[DNS-Error-Reporting] Arends, R. and M. Larson, "DNS Error Reporting", Work in Progress, Internet-Draft, draft-ietf-dnsop-dns-error-reporting-07, 17 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-dns-error-reporting-07>.

[RFC9102] Dukhovni, V., Huque, S., Toorop, W., Wouters, P., and M. Shore, "TLS DNSSEC Chain Extension", RFC 9102, DOI 10.17487/RFC9102, August 2021, <https://www.rfc-editor.org/rfc/rfc9102>.

Appendix A. Early Implementations

[RFC Editor: please remove this section before publication]

This appendix lists some of the implementations of the protocol as it finished working group last call in the DPRIVE Working Group. This list reflects reporting from the DNS community.

- The Unbound resolver has initial experimental code paths to probe over DoT
- The Drink authoritative server supports DoT
- The check-soa tool can probe over DoT
- The Bleau tool can probe over DoT through RIPE Atlas probes
- The PowerDNS Recursor resolver can probe over DoT
- Nameservers for various DNS zones support DoT. These include the root zone (one of the 13 root server identifiers), a social media site, some DNS software developers, and others

Appendix B. Assessing the Experiment

This document is published as an experimental RFC. In order to assess the success of the experiment, some key metrics could be collected by the technical community about the deployment of the protocol in this document. These metrics will be collected in recursive resolvers, authoritative servers, and the networks containing them. Some key metrics include:

- Comparison of the CPU and memory use between Do53 and encrypted transports
- Comparison of the query response rates between Do53 and encrypted transports
- Measurement of server authentication successes and failures
- Measurement and descriptions of observed attack traffic, if any
- Comparison of transactional bandwidth (ingress/egress, packets per second, bytes per second) between Do53 and encrypted transports

Appendix C. Defense Against Active Attackers

The protocol described in this document provides no defense against active attackers. A future protocol for recursive-to-authoritative DNS might want to provide such protection.

This appendix assumes that the use case for that future protocol is a recursive resolver that wants to prevent an active attack on communication between it and an authoritative server that has committed to offering encrypted DNS transport. An inherent part of this use case is that the recursive resolver would want to respond with a `SERVFAIL` response to its client if it cannot make an authenticated encrypted connection to any of the authoritative nameservers for a name.

However, an authoritative server that merely offers encrypted transport (for example, by following the guidance in Section 3) has made no such commitment, and no recursive resolver that prioritizes delivery of DNS records to its clients would want to "fail closed" unilaterally.

So such a future protocol would need at least three major distinctions from the protocol described in this document:

- A signaling mechanism that tells the recursive resolver that the authoritative server intends to offer authenticated encryption
- Authentication of the authoritative server
- A way to combine defense against an active attacker with the defenses described in this document

This can be thought of as a DNS analog to [MTA-STS] or [DANE-SMTP].

C.1. Signaling Mechanism Properties

To defend against an active attacker, the signaling mechanism needs to be able to indicate that the recursive resolver should "fail closed" if it cannot authenticate the server for a particular query.

The signaling mechanism itself would have to be resistant to downgrade attacks from active attackers.

One open question is how such a signal should be scoped. While this document scopes opportunistic state about encrypted transport based on the IP addresses of the client and server, signaled intent to offer encrypted transport is more likely to be scoped by queried zone in the DNS, or by nameserver name than by IP address.

A reasonable authoritative server operator or zone administrator probably doesn't want to risk breaking anything when they first enable the signal. Therefore, a signaling mechanism should probably also offer a means to report problems to the authoritative server operator without the client failing closed. Such a mechanism is likely to be similar to [TLSRPT] or [DNS-Error-Reporting].

C.2. Authentication of Authoritative Server

Forms of server authentication might include:

- an X.509 Certificate issued by a widely-known certification authority associated with the common NS names used for this authoritative server

- DANE authentication (to avoid infinite recursion, the DNS records necessary to authenticate could be transmitted in the TLS handshake using the DNSSEC Chain Extension (see [RFC9102]))

A recursive resolver would have to verify the server's identity. When doing so, the identity would presumably be based on the NS name used for a given query or the IP address of the server.

C.3. Combining Protocols

If this protocol gains reasonable adoption, and a newer protocol that can offer defense against an active attacker were available, deployment is likely to be staggered and incomplete. This means that an operator that want to maximize confidentiality for their users will want to use both protocols together.

Any new stronger protocol should consider how it interacts with the opportunistic protocol defined here, so that operators are not faced with the choice between widespread opportunistic protection against passive attackers (this document) and more narrowly-targeted protection against active attackers.

Appendix D. Document Considerations

[RFC Editor: please remove this section before publication]

D.1. Document History

D.1.1. Substantive Changes from -12 to -13

- Changes based on IESG review

D.1.2. Substantive Changes from -11 to -12

- Editorial changes received during IETF Last Call

D.1.3. Substantive Changes from -10 to -11

- Editorial changes to prepare for IETF Last Call

D.1.4. Substantive Changes from -09 to -10

- Responded to AD review from Eric Vyncke

D.1.5. Substantive Changes from -08 to -09

- Added section with metrics for assessing the experiment
- Updated the definition of unsuccessful responses to encrypted queries

D.1.6. Substantive Changes from -07 to -08

- Changed status to Experimental
- Added operational considerations section
- Many many editorial updates

D.1.7. Substantive Changes from -06 to -07

- Updated how to handle responses from encrypted transports that are slower that Do53

D.1.8. Substantive Changes from -05 to -06

- Clarified the optinality of the protocol
- Spelled out the current scope of DoT and DoQ
- Clarified that responses must be the same on all transports
- Loosened requirement for the resolver to know all its addresses
- Changed examples of unsuccessful responses to timeouts and connection failed
- Expanded acknowledgements
- Added preliminary implementation status

D.1.9. Substantive Changes from -03 to -04

- Clarified that "completed handshake" in Table 2 also includes resumed sessions.
- In Section 4.6.1, specified not to use Do53 even when the last successful connection is far in the past.
- In Section 4.6.3, clarified that an established connection in the established state should not be resumed prematurely.

D.1.10. Substantive Changes from -02 to -03

- Added an Additional Tuning section on recursive resolvers recording other data about an authoritative server's capabilities for future interactions (thank you again Sara Dickinson!). Feedback from operators on how the extra information would be used by a recursive resolver that retains such an expanded state table is particularly welcome.
- Added more text about sharing session state information.
- Added section on modelling the probability of encryption as a future task.

D.1.11. Substantive Changes from -01 to -02

- Removed EDNS Client Subnet recommendations from the probing policy: recursive resolvers implementing the guidance provided in this draft intend to enhance privacy for their users' queries, and although ECS is a valuable feature, it represents a privacy risk. Therefore, a future document encompassing a "how to add privacy" approach could serve for better discussion on this discrepancy (thank you Puneet Sood!).
- Added text on padding queries and responses to mitigate traffic analysis (thank you Sara Dickinson!).
- Put draft on standards track

D.1.12. Substantive Changes from -00 to -01

- Moved discussion of non-opportunistic encryption to an appendix
- Clarify state transitions when sending over encrypted transport

- Introduced new field `E-last-response[X]` for comparison with persistence

D.1.13. Substantive Changes from -01 to -02 (now draft-ietf-dprive-unilateral-probing-00)

- Clarify that deployment to a pool does not need to be strictly simultaneous
- Explain why authoritatives need to serve the same records regardless of SNI
- Defer to external, protocol-specific references for resource management
- Clarify that probed connections must not fail due to authentication failure

D.1.14. draft-dkgjsal-dprive-unilateral-probing Substantive Changes from -00 to -01

- Fallback to cleartext when encrypted transport fails.
- Reduce default `timeout` to 4s
- Clarify SNI guidance: OK for selecting server credentials, not OK for changing answers
- Document ALPN and port numbers
- Justify sorting recursive resolver state by authoritative IP address

Authors' Addresses

Daniel Kahn Gillmor (EDITOR)
American Civil Liberties Union
125 Broad St.
New York, NY, 10004
United States of America
Email: dkg@fifthhorseman.net

Joey Salazar (EDITOR)
Alajuela
20201
Costa Rica
Email: joeygsal@gmail.com

Paul Hoffman (EDITOR)
ICANN
United States of America
Email: paul.hoffman@icann.org