Workgroup:	dprive
Internet-Draft:	draft-ietf-dprive-unilateral-probing-latest
Published:	6 June 2022
Intended Status:	Informational
Expires:	8 December 2022
Authors:	D. K. Gillmor, Ed.	J. Salazar, Ed.	P. Hoffman, Ed.
---	---	---	---
	ACLU		*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. With wider easy deployment of the underlying transport on an opportunistic basis, we hope to 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/.

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 8 December 2022.

Copyright Notice

Table of Contents

1. Introduction

This document aims to provide guidance to implementers 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 A.

The protocol described here offers three concrete advantages to the Internet 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 deployment without external coordination with any of the other parties

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

DoQ: DNS-over-QUIC ([I-D.ietf-dprive-dnsoquic])

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

DoH: DNS-over-HTTPS ([RFC8484])

Encrypted transports: DoQ, DoT, and DoH collectively

2. Priorities

2.1. Minimizing Negative Impacts

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 we specifically try 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.

This document does not pursue the use of DoH 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 resolver to predict the path on its own, in an opportunistic or unilateral fashion, without incurring in excessive use of resources. For instance, a recursive resolver in theory could guess the full path to a queried IP address by trying all the URL paths that the client has in records and see if one of those works, but even though it can be expected that this would work 99% of the time with fewer than 100 probes, this technique would likely incur in excessive resource consumption potentially leading to vulnerabilities and amplification attacks. The authors of this document particularly welcome ideas and contributions from the community that lead to a suitable mechanism for unilaterally probing for DoH-capable authoritative servers, for later consideration in this or other documents.

3. Guidance for Authoritative Servers

An authoritative server **SHOULD** implement and deploy DNS-over-TLS (DoT) on TCP port 853.

An authoritative server **SHOULD** implement and deploy DNS-over-QUIC (DoQ) on UDP port 853.

An authoritative server implementing the protocol described in this document **MUST** implement at least one of DoT or DoQ on port 853.

3.1. Pooled Authoritative Servers Behind a Single IP Address

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 that interacts with such a pool likely does not know that it is a pool. If some members of the pool follow this guidance 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** either:

- ensure that all members of the pool enable the same encrypted transport(s) within the span of a few seconds, or
- ensure that the load balancer maps client requests to pool members based on client IP addresses.

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.

3.2. Authentication

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

The simplest deployment would simply provide a self-issued, regularly-updated X.509 certificate. This mechanism is supported by many TLS and QUIC clients, and will be acceptable for any opportunistic connection.

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, as a probing recursive resolver that offers SNI might or might not have used the right server name to get the records it's 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 6.5 of [I-D.ietf-dprive-dnsoquic] ("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, a reasonable prioritization scheme would be to close connections in this order, until resources are back in control:

- connections with no outstanding queries, ordered by idle time (longest idle time gets closed first)
- 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.

3.4.1. 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. For example, an implementation might use EDNS(0) padding [RFC7830] within an encrypted transport, or a DoQ client might make use of the PADDING frames found in Section 19.1 of [QUIC]). 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

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 delay.

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 recently-good encrypted transport protocol.

If the encrypted transport protocol fails, the recursive resolver falls back to Do53 for that tuple. 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.

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

4.2. Overall Recursive Resolver Settings

A recursive resolver implementing 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 `E-foo` to refer to the `foo` parameter for the encrypted transport `E`.

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`.

This document also assumes that the 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 report their preferred values of these parameters.

4.3. 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 DNS-over-TLS (DoT). A recursive resolver **SHOULD** implement the client side of DNS-over-QUIC (DoQ).

DoT queries from the recursive resolver **MUST** target TCP port 853, with an ALPN of "`dot`". DoQ queries from the recursive resolver **MUST** target UDP port 853, with an ALPN of "`doq`". ALPN is described in [RFC7301].

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 (current common transports are DoH or DoT).

4.4. 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.

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

Name	Description	Retain Across Reset
`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	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 connection	yes

Name	Description	Retain Across Reset
`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.

4.4.1. Separate State for Each of the Recursive Resolver's Own IP Addresses

Note that the recursive resolver should record this per-authoritative-IP state for each 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 `192.0.2.100` and `192.0.2.200`, it should keep two distinct sets of per-authoritative-IP state, one for each source address it uses. 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).

4.5. 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.

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 port closed message).

By associating state with the IP address, the recursive client is most able to avoid reaching a heterogeneous deployment.

For example, consider an authoritative server named `ns0.example.com` that is served by two installations (with two `A` records), one at `192.0.2.7` that follows this guidance, and one at `192.0.2.8` that is a legacy (cleartext port 53-only) deployment. A recursive client who associates state with the `NS` name and reaches `.7` first will "learn" that `ns0.example.com` supports encrypted transport. A subsequent query over encrypted transport dispatched to `.8` would fail, potentially delaying the response.

By associating the state with the authoritative IP address, the client can minimize the number of accidental delays introduced (see also Section 4.4.1 and Section 3.1).

4.6. Probing Policy

When a recursive resolver discovers the need for an authoritative lookup to an authoritative DNS server using IP address `X`, it retrieves the records associated with `X` from its cache.

The following sections 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 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 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`

Otherwise, if there is no outstanding session for any encrypted transport, and the last successful encrypted transport connection was long ago, the 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 a response `R` 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]`:

- Discard `R` and process it no further (do not respond to a cleartext response to a query that is not outstanding)

Otherwise:

- Remove `Q` from `Do53-queries[X]`

If `R` is successful:

- If `Q` is in `Do53-queries[X]`:
  - Return `R` to the requesting client
- For each supported encrypted transport `E`:
  - If `Q` is in `E-queries[X]`:
    - Remove `Q` from `E-queries[X]`

But if `R` is unsuccessful (e.g. `SERVFAIL`):

- if `Q` is not in any of `*-queries[X]`:
  - Return `SERVFAIL` to the client

FIXME: What response should be sent to the client in the case that an extended DNS error ([RFC8914]) is offered in an authoritative's response?

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 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 has a preferred encrypted transport, but only a different transport is in the `established` state, it **MAY** also initiate a new connection to `X` over its preferred transport while concurrently sending the query over the `established` transport `E`.

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

- if `E-session[X]` is in state `pending`, and
- `T0 - 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 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 `fail` or `timeout` and `(T0 - 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 resolver should take the following steps:

- set `E-initiated[X]` to `T0`
- 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 store "early data" from the client into the initial Client Hello in some contexts. A resolver that initiates a connection over a 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, it is **RECOMMENDED** that it implements Encrypted Client Hello ([I-D.ietf-tls-esni]) to 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 `T1`, the resolver sets `E-completed[X]` to `T1` 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 `T1`
- set `E-completed[X]` to `T1`
- 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 `T2` 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 `T2`
- for each query `Q` in `E-queries[X]`:
  - if `Q` is not present in any other `*-queries[X]` or in `Do53-queries[X]`, add `Q` to `Do53-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 `*-queries[X]` or in `Do53-queries[X]`, add `Q` to `Do53-queries[X]` and send query `Q` to `X` over Do53. FIXME: should a resumption ticket be used here for this previously successful connection?

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.

FIXME: are there specific forms of failure that we might handle differently? For example, What if a TCP timeout closes an idle DoT connection? What if a QUIC stream ends up timing out but other streams on the same QUIC connection are going through? Do the described scenarios cover the case when an encrypted transport's port is made unavailable/closed?

4.6.7. Handling Clean Shutdown of an Encrypted Transport Connection

At time `T3`, 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 `*-queries[X]` or in `Do53-queries[X]`, add `Q` to `Do53-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 `T4`, 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 `T4`.

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

4.6.8.1. Avoid EDNS Client Subnet

To protect the privacy of the client, the recursive resolver **SHOULD NOT** send EDNS(0) Client Subnet information to the authoritative server ([RFC7871]) unless explicitly authorized to do so by the client.

4.6.8.2. 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. For example, an implementation might use EDNS(0) padding [RFC7830] within an encrypted transport, or a DoQ client might make use of the PADDING frames found in Section 19.1 of [QUIC]). How much to pad is out of scope of this document, but a reasonable suggestion can be found in [RFC8467].

4.6.8.3. Send Queries in Separate Channels

When multiple queries are multiplexed on a single encrypted transport to a single authoritative server, the recursive resolver **MUST** separate queries clearly and be capable of receiving responses out of order. For guidance on how to best achieve this on a given encrypted transport, see [RFC7766] (for DoT) and [I-D.ietf-dprive-dnsoquic] (for DoQ).

To the extent that the encrypted transport can avoid head-of-line blocking (e.g. QUIC can use a separate stream per query) the recursive resolver **SHOULD** avoid head-of-line blocking.

4.6.9. Receiving a Response over Encrypted Transport

When a response `R` for query `Q` arrives at the recursive resolver over encrypted transport `E` from authoritative server with IP address `X` at time `T5`, the recursive resolver should:

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

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

Otherwise:

- Remove `Q` from `E-queries[X]`
- Set `E-last-activity[X]` to `T5`
- Set `E-last-response[X]` to `T5`

If `R` is successful:

- Return `R` to the requesting client
- For each supported encrypted transport `N` other than `E`:
  - If `Q` is in `N-queries[X]`:
    - Remove `Q` from `N-queries[X]`
- If `Q` is in `Do53-queries[X]`:
  - Remove `Q` from `Do53-queries[X]`

But if `R` is unsuccessful (e.g. `SERVFAIL`):

- If `Q` is not in `Do53-queries[X]` or in any of `*-queries[X]`:
  - Return `SERVFAIL` to the requesting client

FIXME: What response should be sent to the client in the case that an extended DNS error ([RFC8914]) is offered in an authoritative's response?

4.6.10. Resource Exhaustion

To keep resources under control, a recursive resolver should proactively manage outstanding encrypted connections. Section 6.5 of [I-D.ietf-dprive-dnsoquic] ("Connection Handling") 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 resolver to keep a 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.

5. IANA Considerations

IANA does not need to do anything for implementers to adopt the guidance found in this document.

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 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 the all-cleartext status quo at the time of this document.

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 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 signalling 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. Acknowledgements

Many people contributed to the development of this document beyond the authors, including Alexander Mayrhofer, Brian Dickson, Christian Huitema, Eric Nygren, Jim Reid, Kris Shrishak, Ralf Weber, Robert Evans, Sara Dickinson, and the DPRIVE working group.

9. References

9.1. 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/info/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/info/rfc8174>.

[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>.

9.2. 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/info/rfc7435>.

[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>.

[I-D.ietf-dprive-dnsoquic] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-ietf-dprive-dnsoquic-12, 20 April 2022, <https://www.ietf.org/archive/id/draft-ietf-dprive-dnsoquic-12.txt>.

[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>.

[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>.

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

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[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>.

[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/info/rfc8305>.

[RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <https://www.rfc-editor.org/info/rfc8914>.

[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-14, 13 February 2022, <https://www.ietf.org/archive/id/draft-ietf-tls-esni-14.txt>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[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>.

[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/info/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/info/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/info/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/info/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/info/rfc8460>.

[DNS-Error-Reporting] Arends, R. and M. Larson, "DNS Error Reporting", Work in Progress, Internet-Draft, draft-ietf-dnsop-dns-error-reporting-01, 9 November 2021, <https://www.ietf.org/archive/id/draft-ietf-dnsop-dns-error-reporting-01.txt>.

[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/info/rfc9102>.

Appendix A. 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 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].

A.1. Signalling Mechanism Properties

To defend against an active attacker, the signalling 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 signalling 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, signalled 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 signalling 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].

A.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.

A.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 B. Document Considerations

[RFC Editor: please remove this section before publication]

B.1. Document History

B.1.1. Substantive Changes from -01 to -02

- 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
- Moved discussion of non-opportunistic encryption to an appendix

B.1.2. 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
Email: paul.hoffman@icann.org