Unilateral Opportunistic Deployment of Encrypted Recursive-to-Authoritative DNS
draft-ietf-dprive-unilateral-probing-00

Abstract

This draft 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 draft can be defeated by an active attacker, but should be simpler and less risky to deploy than more powerful defenses. The draft also introduces (but does not try to specify) the semantics of signalling that would permit defense against an active attacker.

The goal of this draft 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.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.
Table of Contents

1. Introduction .................................................. 3
   1.1. Requirements Language .................................. 3
   1.2. Terminology ............................................... 3
2. Priorities ...................................................... 3
   2.1. Minimizing Negative Impacts .............................. 4
   2.2. Protocol Choices .......................................... 4
3. Guidance for Authoritative Servers ............................ 5
   3.1. Pooled Authoritative Servers Behind a Single IP Address ........................................ 5
   3.2. Authentication ............................................. 5
   3.3. Server Name Indication .................................... 6
   3.4. Resource Exhaustion ....................................... 6
4. Guidance for recursive resolvers .............................. 7
   4.1. Overall recursive resolver Settings ..................... 7
   4.2. Recursive Resolver Requirements .......................... 8
   4.3. Authoritative Server Encrypted Transport Connection State ........................................... 8
      4.3.1. Separate State for Each of the Recursive Resolver’s Own IP Addresses ............................... 10
   4.4. Maintaining Authoritative State by IP Address .......... 10
   4.5. Probing Policy .............................................. 11
      4.5.1. Sending a query over Do53 ............................ 11
      4.5.2. Receiving a response over Do53 ...................... 12
      4.5.3. Initiating a connection over encrypted transport ........ 12
      4.5.4. Establishing an encrypted transport connection ........ 14
      4.5.5. Failing to establish an encrypted transport connection ................................................. 15
      4.5.6. Encrypted transport failure ......................... 15
      4.5.7. Handling clean shutdown of encrypted transport connection .............................................. 16
      4.5.8. Sending a query over encrypted transport .......... 17
      4.5.9. Receiving a response over encrypted transport .... 17
1.  Introduction

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] and [RFC8174]) when, and only when, they appear in all capitals, as shown here.

1.2.  Terminology

    * "unilateral" means capable of opportunistic probing deployment without external coordination with any of the other parties
    * Do53 refers to traditional cleartext DNS over port 53 ([RFC1035])
    * DoQ refers to DNS-over-QUIC ([I-D.ietf-dprive-dnoquic])
    * DoT refers to DNS-over-TLS ([RFC7858])
    * DoH refers to DNS-over-HTTPS ([RFC8484])
    * Encrypted transports refers to DoQ, DoT, and DoH collectively

2.  Priorities

    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.

2.1. Minimizing Negative Impacts

It also 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 in the "second hump" of the DNS camel, and for uninvolved third parties. The negative impacts that we specifically try to minimize are:

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

2.2. Protocol Choices

While this document focuses specifically on strategies used by DNS servers, it does not go into detail on the specific protocols used, as 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 draft 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 drafts.
3. Guidance for Authoritative Servers

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

An authoritative server MAY implement and deploy DNS-over-QUIC (DoQ) on UDP 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 are updated to follow this guidance while others are 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 heterogenous 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.

Possible alternate forms of server authentication 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 (potentially including the TLS handshake)

### 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 unforseen 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.

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. Overall recursive resolver Settings

A recursive resolver implementing this draft must 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.

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

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 resolver’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.

4.2. 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 MAY 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.

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).

4.3. 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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Retain Across Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>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</td>
<td>N</td>
</tr>
<tr>
<td>initiated</td>
<td>Timestamp of most recent connection attempt</td>
<td>Y</td>
</tr>
<tr>
<td>completed</td>
<td>Timestamp of most recent completed handshake</td>
<td>Y</td>
</tr>
<tr>
<td>status</td>
<td>Enumerated value of success or fail or timeout, associated with the completed handshake</td>
<td>Y</td>
</tr>
<tr>
<td>resumptions</td>
<td>A stack of resumption tickets (and associated parameters) that could be used to resume a prior successful connection</td>
<td>Y</td>
</tr>
<tr>
<td>queries</td>
<td>A queue of queries intended for this authoritative server, each of which has additional status early, unsent, or sent</td>
<td>N</td>
</tr>
<tr>
<td>last-activity</td>
<td>A timestamp of the most recent activity on the connection</td>
<td>N</td>
</tr>
</tbody>
</table>

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.

4.3.1. 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).

4.4. 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 draft 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 heterogenous 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 4.3.1 and Section 3.1).

4.5. 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 know to 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 corresponding on the relevant transport.

4.5.1. 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 (T - E-completed[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].
4.5.2. Receiving a response over Do53

When a successful response $R$ is received in cleartext from authoritative server $X$ for a query $Q$ that was sent over Do53, the recursive resolver should:

* If $Q$ is in Do53-queries[$X$]:
  - Return $R$ to the requesting client
* Remove $Q$ from Do53-queries[$X$]
* For each supported encrypted transport $E$:
  - If $Q$ is in $E$-queries[$X$]:
    o Remove $Q$ from $E$-queries[$X$]
But if $R$ is unsuccessful (e.g. SERVFAIL):

* If $Q$ is in Do53-queries[$X$]:
  - Remove $Q$ from Do53-queries[$X$]
* if $Q$ is not in any of *-queries[$X$]:
  - Return SERVFAIL to the client

4.5.3. Initiating a connection over encrypted transport

If any E-session[$X$] is in the established, the recursive resolver SHOULD NOT initiate a new connection to $X$ over any other transport, but should instead send a query through the existing session (see Section 4.5.8). FIXME: What if there’s a preferred transport, but the established session does not correspond to that preferred transport?

Otherwise, 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
* $T - E$-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 (T - 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).

4.5.3.1. 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 4.5.8.

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

4.5.3.2. 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].
4.5.3.3. 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.

4.5.3.4. Authoritative Server Authentication

A recursive resolver following this guidance MAY attempt to verify the server’s identity by X.509 certificate or DANE. When doing so, the identity would presumably be based on the NS name used for a given query.

However, since 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.

4.5.4. 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

* for each query Q in E-queries[X]:

  - if early data was accepted and Q is early,
    
    o set the status of Q to sent

  - otherwise:

    o send Q through the session (see Section 4.5.8), and set the status of Q to sent

* set E-last-activity[X] to T1

4.5.5. 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.

4.5.6. 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?

4.5.7. Handling clean shutdown of 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.
4.5.8. 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.

When sending query Q, the recursive resolver should ensure that its 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 following guidance:

4.5.8.1. 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.

4.5.8.2. Pad to standard policy

To increase the anonymity set for each query, the recursive resolver SHOULD use EDNS(0) padding according to policies described in [RFC8467].

4.5.8.3. Send queries in separate channels

When multiple queries are multiplexed on a single encrypted transport to a single authoritative server, the recursive resolver MUST offer distinct query ID fields for every outstanding query on a connection, and MUST be capable of receiving responses out of order.

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.

4.5.9. 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, if Q is in E-queries[X], the recursive resolver takes the following steps:

* Remove R from E-queries[X]
* Set E-last-activity[X] to T5
* If R is successful:
  - send R to the requesting client
  - For each supported encrypted transport N other than E:
    o If Q is in N-queries[X]:
      + Remove Q from N-queries[X]
    - If Q is in Do53-queries[X]:
      o Remove Q from Do53-queries[X]
* Otherwise (R is unsuccessful, e.g., SERVFAIL):
  - If Q is not in Do53-queries[X] or any other *-queries[X]:
    o Return SERVFAIL to the requesting client

FIXME: What response should be sent to the clients in the case that extended DNS errors are used in an authoritative’s response?

4.5.10. 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.
4.5.11. 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 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.

5. Signalling for Stronger Defense

This draft _does not_ contemplate the specification of any form of coordinated signalling between authoritative servers and recursive resolvers, as such measures would not be unilateral.

However, the draft highlights the needs of a signaling mechanism for stronger defense.

We highlight the following questions for other specifications to solve:

* What does the signal need to contain?
  - type of transport? (DoQ? DoT? DoH?)
  - error reporting if secure, authenticated connection fails (how to report? similar to TLSRPT?)
  - whether to hard-fail if encrypted communication isn’t available
  - cryptographic authentication of authoritative server (e.g. pubkeys) vs. names vs. domain?

* How should the signal be presented?
  - SVCB RR or "surprising" DS RR

* How should the signal be scoped?
  - per-nameserver (by NS), per-nameserver (by IP address, via in-addr.arpa), or per-domain?
5.1. Combining Signals with Opportunistic Probing

FIXME: How do the signals get combined with the above opportunistic
probing policy? Can we specify that without needing to specify the
signalling mechanism itself?

6. IANA Considerations

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

7. Privacy Considerations

7.1. 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 ECH 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.

8. Security Considerations

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

However, implementers should not rely on the guidance in this draft
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 draft.

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, local root zone, etc, to reduce the overall leakage of query information that could infringe on the client’s privacy.

9. Acknowledgements

Many people contributed to the development of this draft beyond the authors, including Brian Dickson, Christian Huitema, Eric Nygren, Jim Reid, Kris Shrishak, Paul Hoffman, Ralf Weber, Robert Evans, and the DPRIVE working group.

10. References

10.1. Normative References


10.2. Informative References


[I-D.ietf-dprive-dnsoquic]

[I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft,


Appendix A. Document Considerations

[ RFC Editor: please remove this section before publication ]

This document is currently edited as markdown. Minor editorial changes can be suggested via merge requests at https://gitlab.com/dkg/dprive-unilateral-probing or by e-mail to the editor. Please direct all significant commentary to the public IETF DPRIVE mailing list: dprive@ietf.org

A.1. Document History

A.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

A.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
American Civil Liberties Union
125 Broad St.
New York, NY, 10004
United States of America
Email: dkg@fifthhorseman.net

Joey Salazar
Alajuela
20201
Costa Rica
Email: joeygsal@gmail.com
Network policy to use Network-designated DNS Resolvers
draft-reddy-add-enterprise-policy-01

Abstract

This document specifies a mechanism to inform endpoints about any network policy mandating the use of network-designated DNS resolvers.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
Table of Contents

1. Introduction ................................. 2
2. Terminology .................................. 3
3. PvD NetworkDNSOnly and ErrorNetworkDNSOnly Keys ....................... 4
4. Scope of NetworkDNSOnly Key ................................ 5
5. An Example .................................... 7
6. Security Considerations ................................ 7
7. IANA Considerations ................................ 7
8. Acknowledgements ................................ 7
9. References ...................................... 7
   9.1. Normative References ........................ 8
   9.2. Informative References ...................... 8
Authors’ Addresses .................................. 10

1. Introduction

Historically, an endpoint would utilize network-designated DNS servers upon joining a network (e.g., DHCP OFFER, IPv6 Router Advertisement). While it has long been possible to configure endpoints to ignore the network’s suggestions and use a (public) DNS server on the Internet, this was seldom used because some networks block UDP/53 (in order to enforce their own DNS policies). Also, there has been an increase in the availability of "public resolvers" [RFC8499] which DNS clients may be pre-configured to use instead of the default network resolver for a variety of reasons (e.g., offer a good reachability, support an encrypted transport, provide a claimed privacy policy, (lack of) filtering). With the advent of DoT and DoH, such network blocking is more difficult. The network is unable to express its policy to use network-designated resolvers to the endpoints and the endpoint is unable to identify the reason why the public DNS server is not reachable.

DNS resolvers not signaled by the network (e.g., DNS-over-TLS (DoT) [RFC7858] or DNS-over-HTTPS (DoH) [RFC8484]) will bypass enterprise-specific policies, including security policies for endpoints (e.g., laptops, printers, IoT devices). It is out of the scope of this memo to characterize such policies nor assess whether they achieve the claimed intent.

With the advent of DoT and DoH, the network is unable to express any policy to the endpoints to explain why the network is blocking alternative resolvers, and endpoints are unable to identify the reason why their choice of public DNS resolver is not reachable. Although network security services can be configured to block DoT traffic by dropping outgoing packets to destination port number 853, identifying DoH traffic is more challenging: network security services may try to identify the well-known DoH resolvers by their
domain name and DoH traffic can be blocked by dropping outgoing packets to these domains. However, DoH traffic can not be fully identified without acting as a TLS proxy, with potentially many undesired consequences.

This results in incompatibilities with the privacy profiles discussed in [RFC8310]:

* If an endpoint has enabled strict privacy profile (Section 5 of [RFC8310]), the endpoint cannot resolve DNS names.

* If an endpoint has enabled opportunistic privacy profile (Section 5 of [RFC8310]), the endpoint will either fallback to an encrypted connection without authenticating the DNS server signaled by the local network or fallback to clear text DNS, and cannot exchange encrypted DNS messages.

The fallback adversely impacts security and privacy as internal attacks are possible within Enterprise networks. For example, an internal attacker can modify the DNS responses to re-direct a client to malicious servers or pervasively monitor the DNS traffic.

This document describes a mechanism for informing endpoints of network policy related to network-designated DNS servers, such as those DNS servers signaled using [I-D.ietf-add-dnr] and [I-D.ietf-add-ddr].

2. Terminology

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.

This document makes use of the terms defined in [RFC8499]. The terms "Private DNS", "Global DNS" and "Split DNS" are defined in [RFC8499].

'Encrypted DNS' refers to a DNS protocol that provides an encrypted channel between a DNS client and server (e.g., DoT, DoH, or DoQ).

The term "enterprise network" in this document extends to a wide variety of deployment scenarios. For example, an "enterprise" can be a Small Office, Home Office, Corporation, Education facility. The clients that connect to an enterprise network can securely authenticate that network and the client is sure that it has connected to the network it was expecting to.
3. PvD NetworkDNSOnly and ErrorNetworkDNSOnly Keys

Provisioning Domains (PvDs) are defined in [RFC7556] as sets of network configuration information that clients can use to access networks, including rules for DNS resolution and proxy configuration. [RFC8801] defines a mechanism for discovering multiple Explicit PvDs on a single network and their Additional Information by means of an HTTP-over-TLS query using a URI derived from the PvD ID. This set of additional configuration information is referred to as a Web Provisioning Domain (Web PvD).

This document defines two PvD Key:

The NetworkDNSOnly PvD Key: which determines if network will block, or attempt to block, DNS queries sent to DNS servers that were not signaled by the network. This key has the value True or False (case insensitive).

The ErrorNetworkDNSOnly PvD Key: which contains an extended DNS error code as defined by [RFC8914]. This key is only present if NetworkDNSOnly is True.

Where enterprise networks require clients to query the network-designated DNS servers, it sets the PvD NetworkDNSOnly key to True, otherwise sets NetworkDNSOnly to False. If NetworkDNSOnly is set to True, it implies the network will block, or attempt to block, DNS queries sent to DNS servers that were not signaled by the network. If NetworkDNSOnly is True, the ErrorNetworkDNSOnly key MUST contain either 15, 16 or 17 Extended DNS error codes as defined by [RFC8914]. Note that the extended error code "Blocked" defined in Section 4.16 of [RFC8914] identifies indicates that access to domains is blocked due to an policy by the operator of the DNS server, extended error code "Censored" defined in Section 4.17 of [RFC8914] identifies access to domains is blocked based on a requirement from an external entity and the extended error code "Filtered" defined in Section 4.18 of [RFC8914] identifies access to domains is blocked based on the request from the client to blacklist domains.
The ErrorNetworkDNSOnly key is useful when the client does not use DNS resolution by the network-designated DNS server to acquire the IP addresses of alternate DNS servers. For example, the client can be pre-configured with both the domain name and IP addresses of the DNS server not signaled by the network (Section 7.1 in [RFC8310]) or the client can be pre-configured with the IP address of the resolver, and it uses IP address in the certificate as identifier (see [RFC8738]). Further, the ErrorNetworkDNSOnly key is useful when the network security service fails to block access to non-network DNS server but successfully filters traffic from the endpoint to IP addresses not conveyed to the endpoint as part of DNS resolution by the network-designated DNS server.

NetworkDNSOnly set to True is an internal security policy expression by the operator of the network but is not a policy prescription to the endpoints to disable its use of its other configured DNS servers; that is, the endpoint can ignore NetworkDNSOnly set to True. If joining an un-trusted network (e.g., coffeeshop, hotel, airport network), a True value of NetworkDNSOnly MUST be ignored. The mechanism the client uses to determine 'trusted network' to assist the user MUST involve authenticated identity of the network (not merely matching SSID in the case of WiFi), such as 802.1X or confirming the network-designated encrypted resolver name is pre-configured in the Operating System and TLS handshake with it succeeds. For example, the client can determine "Open" (unencrypted) wireless networks are untrusted networks, notify the user that using a shared and public Pre-Shared Key (PSK) for wireless authentication is an untrusted network. If the pre-shared-key is the same for all clients that connect to the same WLAN, the shared key will be available to all nodes, including attackers, so it is possible to mount an active on-path attack (e.g., [Evil-Twin], [Krack], [Dragonblood]). For example, coffee shops and air ports use PSK and are unwilling to perform complex configuration on their networks. In addition, customers are generally unwilling to do complicated provisioning on their devices just to obtain free Wi-Fi. This type of networks can be tagged as "untrusted networks" with minimal human intervention. In such cases the endpoint MAY choose to use an alternate network (e.g., cellular) to resolve the global domain names.

4. Scope of NetworkDNSOnly Key

If a device is managed by an enterprise’s IT department, the device can be configured to use a specific encrypted DNS server. This configuration may be manual or rely upon whatever deployed device management tool in an enterprise network. For example, customizing Firefox using Group Policy to use the Enterprise DoH server is discussed in [Firefox-Policy] for Windows and MacOS, and setting
Chrome policies is discussed in [Chrome-Policy] and [Chrome-DoH].

If mobile device management (MDM) (e.g., [MDM-Apple]) secures a device, MDM can configure OS/browser with a specific encrypted DNS server. If an endpoint is on-boarded, for example, using Over-The-Air (OTA) enrollment [OTA] to provision the device with a certificate and configuration profile, the configuration profile can include the authentication domain name (ADN) of the encrypted DNS server. The OS/Browser can use the configuration profile to use a specific encrypted DNS server. In this case, MDM is not installed on the device.

Provisioning IT-managed devices, BYOD devices with MDM or configuration profile with network-designated DNS server is outside the scope of this document.

Typically, Enterprise networks do not assume that all devices in their network are managed by the IT team or MDM, especially in the quite common BYOD scenario. The endpoint can use the discovered network-designated DNS server to only access DNS names for which the Enterprise network claims authority and use another public DNS server for global domains or use the discovered network-designated DNS server to access both private domains and global domains.

The scope of NetworkDNSOnly key is restricted to unmanaged BYOD devices without a configuration profile on explicitly trusted networks. In this use case, the user has authorized the client to override local DNS settings for a specific network. It is similar to the way users explicitly disable VPN connection in specific networks and VPN connection is enabled by default in other networks for privacy. The unmanaged BYOD devices use mutual authentication of the client and the enterprise network. The client is typically authenticated with their user credentials (e.g., username and password). The network is typically authenticated with a certificate (e.g., PEAP-MSCHAPv2 [PEAP]) or a mutually-authenticated key exchange which is well-defended from offline attacks (e.g., EAP-pwd [RFC8146], EAP-PSK [RFC4764]). Importantly, WPA-PSK and WPA2-PSK are not well-defended from offline attacks and MUST NOT be used in conjunction with NetworkDNSOnly set to True.

Note: Many users have privacy and personal data sovereignty concerns with employers installing MDM on their personal devices; they are concerned that admin can glean personal information and could control how they use their devices. When users do not install MDM on their devices, IT admins do not get visibility into the security posture of those devices. To overcome this problem, a host agent can cryptographically attest the security status associated with device, such as minimum pass code length,
biometric login enabled, OS version etc. This approach is fast gaining traction especially with the advent of closed OS like Windows 10 in S mode [win10s] or Chromebook [Chromebook], where applications are sandboxed (e.g., ransomware attack is not possible) and applications can only be installed via the OS store.

5. An Example

The following example shows how the JSON keys defined in this document can be used:

```json
{
    "identifier": "cafe.example.com.",
    "expires": "2020-05-23T06:00:00Z",
    "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
    "NetworkDNSOnly": True,
    "ErrorNetworkDNSOnly": 15
}
```

The JSON keys "identifier", "expires", and "prefixes" are defined in [RFC8801].

6. Security Considerations

The content of NetworkDNSOnly and ErrorSplitDNSBlocked may be passed to another (DNS) program for processing. As with any network input, the content SHOULD be considered untrusted and handled accordingly. The security considerations discussed in Section 3 and Section 4 need to be considered to restrict the scope of NetworkDNSOnly and ErrorSplitDNSBlocked PvD Keys to explicitly trusted networks. The NetworkDNSOnly and ErrorSplitDNSBlocked PvD Keys assigned by an anonymous or unknown network (e.g., coffee shops) MUST be ignored by the client.

7. IANA Considerations

IANA is requested to add NetworkDNSOnly and ErrorSplitDNSBlocked PvD Keys to the Additional Information PvD Keys registry (https://www.iana.org/assignments/pvds/pvds.xhtml).

8. Acknowledgements

Thanks to Mohamed Boucadair, Jim Reid, Ben Schwartz, Tommy Pauly, Paul Vixie, Ben Schwartz, and Vinny Parla for the discussion and comments.

9. References
9.1. Normative References


9.2. Informative References


[PEAP] Microsoft, "[MS-PEAP]: Protected Extensible Authentication Protocol (PEAP)", <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-peap/5308642b-90c9-4cc4-beec-fb367325c0f9>


Authors’ Addresses

Tirumaleswar Reddy
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
United States of America
Email: danwing@gmail.com

Kevin Smith
Vodafone Group
One Kingdom Street
London
United Kingdom
Email: kevin.smith@vodafone.com
Abstract

When split-horizon DNS is deployed by a network, certain domains can be resolved authoritatively by the network-provided DNS resolver. DNS clients that don’t always use this resolver might wish to do so for these domains. This specification enables networks to inform DNS clients about domains that are inside the split-horizon DNS, and describes how clients can confirm the local resolver’s authority over these domains.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
To resolve a DNS query, there are three essential behaviors that an implementation can apply: (1) answer from a local database, (2) query the relevant authorities and their parents, or (3) ask a server to query those authorities and return the final answer. Implementations that use these behaviors are called "authoritative nameservers", "full resolvers", and "forwardsers" (or "stub resolvers"). However, an implementation can also implement a mixture of these behaviors, depending on a local policy, for each query. We term such an implementation a "hybrid resolver".
Most DNS resolvers are hybrids of some kind. For example, stub resolvers frequently support a local "hosts file" that preempts query forwarding, and most DNS forwarders and full resolvers can also serve responses from a local zone file. Other standardized hybrid resolution behaviors include Local Root [RFC8806], mDNS [RFC6762], and NXDOMAIN synthesis for .onion [RFC7686].

In many network environments, the network offers clients a DNS server (e.g. DHCP OFFER, IPv6 Router Advertisement). Although this server is formally specified as a recursive resolver (e.g. Section 5.1 of [RFC6106]), some networks provide a hybrid resolver instead. If this resolver acts as an authoritative server for some names, we say that the network has "split-horizon DNS", because those names resolve in this way only from inside the network.

Network clients that use pure stub resolution, sending all queries to the network-provided resolver, will always receive the split-horizon results. Conversely, clients that send all queries to a different resolver or implement pure full resolution locally will never receive them. Clients with either pure resolution behavior are out of scope for this specification. Instead, this specification enables hybrid clients to access split-horizon results from a network-provided hybrid resolver, while using a different resolution method for some or all other names.

To achieve the required security properties, clients must be able to authenticate the DNS servers provided by the network, for example using the techniques proposed in [I-D.ietf-add-dnr] and [I-D.ietf-add-ddr], and prove that they are authorized to serve the offered split-horizon DNS names. As a result, use of this specification is limited to servers that support authenticated encryption and split-horizon DNS names that are properly rooted in the global DNS.

2. Terminology

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.

This document makes use of the terms defined in [RFC8499]. The terms "Private DNS", "Global DNS" and "Split DNS" are defined in [RFC8499].

"Encrypted DNS" refers to a DNS protocol that provides an encrypted channel between a DNS client and server (e.g., DoT, DoH, or DoQ).
The terms ‘Authorized Split Horizon’ and ‘Domain Camping’ are also defined.

2.1. Authorized Split Horizon

A split horizon configuration for some name is considered "authorized" if any parent of that name has given the local network permission to serve its own responses for that name. Such authorizations generally extend to the entire subtree of names below the authorization point.

2.2. Domain Camping

Domain Camping refers to operating a nameserver which claims to be authoritative for a zone, but actually isn’t. For example, a domain called example.com on the Internet and an internal DNS server also claims to be authoritative for example.com, but has no delegation from example.com on the Internet. Someone might domain camp on a popular domain name providing the ability to monitor queries and modify answers for that domain.

A common variation on domain camping is "NXDOMAIN camping", in which a nameserver claims a zone that does not exist in the global DNS. This is a form of domain camping because it seizes a portion of the parent zone without permission. The use of nonexistent TLDs for local services is a form of NXDOMAIN camping on the root zone.

Any form of domain camping likely violates the IAB’s guidance regarding "the Unique DNS Root" [RFC2826].

3. Scope

The protocol in this document allows the domain owner to create a split-horizon DNS. Other entities which do not own the domain are detected by the client. Thus, DNS filtering is not enabled by this protocol.

4. Provisioning Domains dnsZones

Provisioning Domains (PvDs) are defined in [RFC7556] as sets of network configuration information that clients can use to access networks, including rules for DNS resolution and proxy configuration. The PvD Key dnsZones is defined in [RFC8801]. The PvD Key dnsZones notifies clients of names for which one of the network-provided resolvers is authoritative. Attempting to resolve these names via another resolver might fail or return results that are not correct for this network.
Each dnsZones entry indicates a claim of authority over a domain and its subdomains. For example, if the dnsZones entry is "example.test", this covers "example.test", "www.example.test", and "mail.eng.example.test", but not "otherexample.test" or "example.test.net".

[RFC8801] defines a mechanism for discovering multiple Explicit PvDs on a single network and their Additional Information by means of an HTTP-over-TLS query using a URI derived from the PvD ID. This set of additional configuration information is referred to as a Web Provisioning Domain (Web PvD). The PvD RA option defined in [RFC8801] SHOULD set the H-flag to indicate that Additional Information is available. This Additional Information JSON object SHOULD include the "dnsZones" key to define the DNS domains for which the network claims authority.

4.1. Confirming Authority over the Domains

To comply with [RFC2826], each dnsZones entry must be authorized in the global DNS hierarchy. To prevent domain camping, clients must confirm this authorization before making use of the entry.

To enable confirmation, the client must discover and validate the Authentication Domain Names (ADNs) of the network-designated resolvers using a method such as DNR [I-D.ietf-add-dnr]. The client must also perform an NS query for each dnsZones entry and confirm that at least one of the ADNs appears in each NS RRSet. This NS query must be conducted in a manner that is not vulnerable to tampering by the local network. Suitable tamperproof resolution strategies are described in Section 4.1.1 and Section 4.1.2.

Note that each dnsZones entry is authorized only for the specific resolvers whose ADNs appear in its NS RRSet. If a network offers multiple encrypted resolvers via DNR, each dnsZones entry may be authorized for a distinct subset of the network-provided resolvers.

4.1.1. Confirmation using a pre-configured public resolver

The client sends an NS query for the domain in dnsZones to a pre-configured resolver that is external to the network, over a secure transport. Clients SHOULD apply whatever acceptance rules they would otherwise apply when using this resolver (e.g. checking the AD bit, validating RRSIGs).
4.1.2. Confirmation using DNSSEC

The client resolves the NS record using any resolution method of its choice (e.g. querying one of the network-provided resolvers, performing iterative resolution locally), and performs full DNSSEC validation locally [RFC6698]. The result is processed based on its DNSSEC validation state (Section 4.3 of [RFC4035]):

* "Secure": the NS record is used for confirmation.
* "Bogus" or "Indeterminate": the record is rejected and confirmation is considered to have failed.
* "Insecure": the client SHOULD retry the confirmation process using a different method, such as the one in Section 4.1.1, to ensure compatibility with unsigned names.

5. An example of Split-Horizon DNS Configuration

Consider an organization that operates "example.com", and runs a different version of its global domain on its internal network. Today, on the Internet it publishes two NS records, "ns1.example.com" and "ns2.example.com".

To add support for the mechanism described in this document, the network and endpoints first need to support [I-D.ietf-add-dnr] and [RFC8801]. Then, for each site, the administrator would add DNS servers named "ns1.example.com" or "ns2.example.com" (the names published on the Internet). Those names would be advertised to the endpoints as described in [I-D.ietf-add-dnr].

The endpoints compliant with this specification can then determine the network's internal nameservers are owned and managed by the same entity that has published the NS records on the Internet as shown in Figure 1:

Steps 1-2: The client joins the network, obtains an IP address, and discovers the resolvers "ns1.example.com" and "ns2.example.com" and their IP addresses using DNR [I-D.ietf-add-dnr]. Using [RFC8801], the client also discovers the PvD FQDN is "pvd.example.com".

Steps 3-7: The client establishes an encrypted DNS connection with "ns1.example.com", validates its TLS certificate, and queries it for "pvd.example.com" to retrieve the PvD JSON object. Note that [RFC8801] in Section 4.1 mandates the PvD FQDN MUST be resolved using the DNS servers indicated by the associated PvD. The PvD contains:
The JSON keys "identifier", "expires", and "prefixes" are defined in [RFC8801].

Steps 8-9: The client then uses an encrypted DNS connection to a public resolver (e.g., 1.1.1.1) to issue NS queries for the domains in dnsZones. The NS lookup for "example.com" will return "ns1.example.com" and "ns2.example.com".

Step 10: As the network-provided nameservers are the same as the names retrieved from the public resolver and the network-designated resolver’s certificate includes at least one of the names retrieved from the public resolver, the client has finished validation that the nameservers signaled in [I-D.ietf-add-dnr] and [RFC8801] are owned and managed by the same entity that published the NS records on the Internet. The endpoint will then use that information from [I-D.ietf-add-dnr] and [RFC8801] to resolve names within dnsZones.
| validate TLS certificate | | | |
|--------------------------| | | |
|--------------------------| | | |
| resolve pvd.example.com (4) | | | |
|--------------------------| | | |

<p>| use network-designated resolver                   |</p>
<table>
<thead>
<tr>
<th>for example.com (10)</th>
</tr>
</thead>
</table>

Figure 1: An Example of Split-Horizon DNS Configuration

6. Split DNS Configuration for IKEv2

The split-tunnel Virtual Private Network (VPN) configuration allows the endpoint to access resources that reside in the VPN [RFC8598] via the tunnel; other traffic not destined to the VPN does not traverse the tunnel. In contrast, a non-split-tunnel VPN configuration causes all traffic to traverse the tunnel into the VPN.

When the VPN tunnel is IPsec, the encrypted DNS resolver hosted by the VPN service provider can be securely discovered by the endpoint using the ENCDNS_IP*-* IKEv2 Configuration Payload Attribute Types defined in [I-D.ietf-ipsecme-add-ike]. For split-tunnel VPN
configurations, the endpoint uses the discovered encrypted DNS server to resolve domain names for which the VPN provider claims authority. For non-split-tunnel VPN configurations, the endpoint uses the discovered encrypted DNS server to resolve both global and private domain names. For split-tunnel VPN configurations, the IKE client can use any one of the mechanisms discussed in Section 4.1 to determine if the VPN service provider is authoritative over the Split Horizon DNS domains.

Other VPN tunnel types have similar configuration capabilities, not detailed here.

7.  Security Considerations

The content of dnsZones may be passed to another (DNS) program for processing. As with any network input, the content SHOULD be considered untrusted and handled accordingly. The client must perform the mechanisms discussed in Section 4.1 to determine if the network-designated encrypted resolvers are authoritative over the domains in dnsZones. If they are not, the client must ignore those dnsZones.

This specification does not alter DNSSEC validation behaviour. To ensure compatibility with validating clients, network operators MUST ensure that names under the split horizon are correctly signed or place them in an unsigned zone.

8.  IANA Considerations

This document has no IANA actions.

9.  Acknowledgements

Thanks to Mohamed Boucadair, Jim Reid, Tommy Pauly, Paul Vixie, Paul Wouters and Vinny Parla for the discussion and comments.

10.  References

10.1.  Normative References


10.2. Informative References


Authors’ Addresses

Tirumaleswar Reddy  
Akamai  
Embassy Golf Link Business Park  
Bangalore 560071  
Karnataka  
India  
Email: kondtir@gmail.com

Dan Wing  
Citrix Systems, Inc.  
4988 Great America Pkwy  
Santa Clara, CA 95054  
United States of America  
Email: danwing@gmail.com
Kevin Smith  
Vodafone Group  
One Kingdom Street  
London  
United Kingdom  
Email: kevin.smith@vodafone.com

Benjamin Schwartz  
Google LLC  
Email: bemasc@google.com
Establishing Local DNS Authority in Split-Horizon Environments
draft-reddy-add-enterprise-split-dns-10

Abstract

When split-horizon DNS is deployed by a network, certain domains can be resolved authoritatively by the network-provided DNS resolver. DNS clients that don’t always use this resolver might wish to do so for these domains. This specification describes how clients can confirm the local resolver’s authority over these domains.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights...
1. Introduction

To resolve a DNS query, there are three essential behaviors that an implementation can apply: (1) answer from a local database, (2) query the relevant authorities and their parents, or (3) ask a server to query those authorities and return the final answer. Implementations that use these behaviors are called "authoritative nameservers", "full resolvers", and "forwardsers" (or "stub resolvers"). However, an implementation can also implement a mixture of these behaviors, depending on a local policy, for each query. We term such an implementation a "hybrid resolver".
Most DNS resolvers are hybrids of some kind. For example, stub resolvers frequently support a local "hosts file" that preempts query forwarding, and most DNS forwarders and full resolvers can also serve responses from a local zone file. Other standardized hybrid resolution behaviors include Local Root [RFC8806], mDNS [RFC6762], and NXDOMAIN synthesis for .onion [RFC7686].

In many network environments, the network offers clients a DNS server (e.g. DHCP OFFER, IPv6 Router Advertisement). Although this server is formally specified as a recursive resolver (e.g. Section 5.1 of [RFC6106]), some networks provide a hybrid resolver instead. If this resolver acts as an authoritative server for some names, we say that the network has "split-horizon DNS", because those names resolve in this way only from inside the network.

Network clients that use pure stub resolution, sending all queries to the network-provided resolver, will always receive the split-horizon results. Conversely, clients that send all queries to a different resolver or implement pure full resolution locally will never receive them. Clients with either pure resolution behavior are out of scope for this specification. Instead, this specification enables hybrid clients to access split-horizon results from a network-provided hybrid resolver, while using a different resolution method for some or all other names.

There are several existing mechanisms for a network to provide clients with "local domain hints", listing domain names that have special treatment in this network (Section 4). However, none of the local domain hint mechanisms enable clients to determine whether this special treatment is authorized by the domain owner. Instead, these specifications require clients to make their own determinations about whether to trust and rely on these hints.

This specification describes a protocol between domains, networks, and clients that allows the network to establish its authority over a domain to a client (Section 5). Clients can use this protocol to confirm that a local domain hint was authorized by the domain (Section 6), which might influence its processing of that hint.

This specification relies on securely identified local DNS servers and globally valid NS records. Use of this specification is therefore limited to servers that support authenticated encryption and split-horizon DNS names that are properly rooted in the global DNS.
2. Terminology

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.

This document makes use of the terms defined in [RFC8499]. The term "Global DNS" is defined in [RFC8499].

'Encrypted DNS' refers to a DNS protocol that provides an encrypted channel between a DNS client and server (e.g., DoT, DoH, or DoQ).

The term 'Validated Split-Horizon' is also defined.

2.1. Validated Split-Horizon

A split horizon configuration for some name is considered "validated" if the network client has confirmed that a parent of that name has authorized the local network to serve its own responses for that name. Such authorization generally extends to the entire subtree of names below the authorization point.

3. Scope

The protocol in this document allows the domain owner to create a split-horizon DNS. Other entities which do not own the domain are detected by the client. Thus, DNS filtering is not enabled by this protocol.

4. Local Domain Hint Mechanisms

There are various mechanisms by which a network client might learn "local domain hints", which indicate a special treatment for particular domain names upon joining a network. This section provides a review of some common and standardized mechanisms for receiving domain hints.

4.1. DHCP Options

There are several DHCP options that convey local domain hints of different kinds. The most directly relevant is "RDNSS Selection" [RFC6731], which provides "a list of domains ... about which the RDNSS has special knowledge", along with a "High", "Medium", or "Low" preference for each name. The specification notes the difficulty of relying on these hints without validation:
Trustworthiness of an interface and configuration information received over the interface is implementation and/or node deployment dependent, and the details of determining that trust are beyond the scope of this specification.

Other local domain hints in DHCP include the "Domain Name" [RFC2132], "Access Network Domain Name" [RFC5986], "Client FQDN" [RFC4702][RFC4704], and "Name Service Search" [RFC2937] options. This specification may help clients to interpret these hints. For example, a rogue DHCP server could use the "Client FQDN" option to assign a client the name "www.example.com" in order to prevent the client from reaching the true "www.example.com". A client could use this specification to check the network’s authority over this name, and adjust its behavior to avoid this attack if authority is not established.

The Domain Search option [RFC3397] [RFC3646], which offers clients a way to expand short names into Fully Qualified Domain Names, is not a "local domain hint" by this definition, because it does not modify the processing of any specific domain. (The specification notes that this option can be a "fruitful avenue of attack for a rogue DHCP server", and provides a number of cautions against accepting it unconditionally.)

4.2. Host Configuration

A host can be configured with DNS information when it joins a network, including when it brings up VPN (which is also considered joining a(n additional) network, detailed in Section 8). Existing implementations determine the host has joined a certain network via SSID, IP subnet assigned, DNS server IP address or name, and other similar mechanisms. For example, one existing implementation determines the host has joined an internal network because the DHCP-assigned IP address belongs to the company’s IP address (as assigned by the regional IP addressing authority) and the DHCP-advertised DNS IP address is one used by IT at that network. Other mechanisms exist in other products but are not interesting to this specification; rather what is interesting is this step to determine "we have joined the internal corporate network" occurred and the DNS server is configured as authoritative for certain DNS zones (e.g., *.example.com).

Because a rogue network can simulate all or most of the above characteristics this specification details how to validate these claims in Section 6.
4.3. Provisioning Domains dnsZones

Provisioning Domains (PvDs) are defined in [RFC7556] as sets of network configuration information that clients can use to access networks, including rules for DNS resolution and proxy configuration. The PvD Key "dnsZones" is defined in [RFC8801] as a list of "DNS zones searchable and accessible" in this provisioning domain. Attempting to resolve these names via another resolver might fail or return results that are not correct for this network.

4.4. Split DNS Configuration for IKEv2

In IKEv2 VPNs, the INTERNAL_DNS_DOMAIN configuration attribute can be used to indicate that a domain is "internal" to the VPN [RFC8598]. To prevent abuse, the specification notes various possible restrictions on the use of this attribute:

"If a client is configured by local policy to only accept a limited set of INTERNAL_DNS_DOMAIN values, the client MUST ignore any other INTERNAL_DNS_DOMAIN values."

"IKE clients MAY want to require whitelisted domains for Top-Level Domains (TLDs) and Second-Level Domains (SLDs) to further prevent malicious DNS redirections for well-known domains."

Within these guidelines, a client could adopt a local policy of accepting INTERNAL_DNS_DOMAIN values only when it can validate the local DNS server’s authority over those names as described in this specification.

5. Establishing Local DNS Authority

To establish its authority over some DNS zone, a participating network MUST offer one or more encrypted resolvers via DNR [I-D.ietf-add-dnr] or an equivalent mechanism (see Section 8). At least one of these resolvers’ Authentication Domain Names (ADNs) MUST appear in an NS record for that zone. This arrangement establishes this resolver’s authority over the zone.

6. Validating Authority over Local Domain Hints

To validate the network’s authority over a domain name, participating clients MUST resolve the NS record for that name. If the resolution result is NODATA, the client MUST remove the last label and repeat the query until a response other than NODATA is received.
Once the NS record has been resolved, the client MUST check if each local encrypted resolver’s Authentication Domain Name appears in the NS record. The client SHALL regard each such resolver as authoritative for the zone of this NS record.

Each validation of authority applies only to the specific resolvers whose names appear in the NS RRSet. If a network offers multiple encrypted resolvers, each DNS entry may be authorized for a distinct subset of the network-provided resolvers.

A zone is termed a "Validated Split-Horizon zone" after successful validation using a "tamperproof" NS resolution method, i.e. a method that is not subject to interference by the local network operator. Two possible tamperproof resolution methods are presented below.

6.1. Using Pre-configured Public Resolver

The client sends the NS query to a pre-configured resolver that is external to the network, over a secure transport. Clients SHOULD apply whatever acceptance rules they would otherwise apply when using this resolver (e.g. checking the AD bit, validating RRSIGs).

6.2. Using DNSSEC

The client resolves the NS record using any resolution method of its choice (e.g. querying one of the network-provided resolvers, performing iterative resolution locally), and performs full DNSSEC validation locally [RFC6698]. The result is processed based on its DNSSEC validation state (Section 4.3 of [RFC4035]):

Secure: the response is used for validation.

Bogus or Indeterminate: the response is rejected and validation is considered to have failed.

Insecure: the client SHOULD retry the validation process using a different method, such as the one in Section 6.1, to ensure compatibility with unsigned names.

7. Examples of Split-Horizon DNS Configuration

Two examples are shown below. The first example showing an company with an internal-only DNS server resolving the entire zone for that company (e.g., *.example.com) the second example resolving only a subdomain of the company’s zone (e.g., *.internal.example.com).
7.1. Split-Horizon Entire Zone

Consider an organization that operates "example.com", and runs a different version of its global domain on its internal network. Today, on the Internet it publishes two NS records, "ns1.example.com" and "ns2.example.com".

The host and network first need mutual support one of the mechanisms described in learning (Section 4). Shown in Figure 1 is learning using DNR and PvD.

Validation is then performed using either Public DNS (Section 7.1.1) or DNSSEC (Section 7.1.2).

steps 1-2: The client determines the network’s DNS server (ns1.example.com) and Provisioning Domain (pvd.example.com) using DNR [I-D.ietf-add-dnr] and PvD [RFC8801], using one of DNR Router Solicitation, DHCPv4, or DHCPv6.

step 3-5: The client connects to the DNR-learned DNS server (ns1.example.com), validates its certificate, and queries for pvd.example.com.

steps 6-7: The client connects to the PvD server, validates its certificate, and retrieves the provisioning domain JSON information indicated by the associated PvD. The PvD contains:

```
{
  "identifier": "pvd.example.com",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "dnsZones": ["example.com"]
}
```

The JSON keys "identifier", "expires", and "prefixes" are defined in [RFC8801].
7.1.1. Verification using Public Resolver

The figure below shows the steps performed to verify the local claims of DNS authority using a public resolver.

Steps 1-2: The client uses an encrypted DNS connection to a public resolver.

---

**Figure 1: Learning Local Claims of DNS Authority**
resolver (e.g., 1.1.1.1) to issue NS queries for the domains in dnsZones. The NS lookup for "example.com" will return "ns1.example.com" and "ns2.example.com".

Step 3: As the network-provided nameservers are the same as the names retrieved from the public resolver and the network-designated resolver’s certificate includes at least one of the names retrieved from the public resolver, the client has finished validation that the nameservers signaled in [I-D.ietf-add-dnr] and [RFC8801] are owned and managed by the same entity that published the NS records on the Internet. The endpoint will then use that information from [I-D.ietf-add-dnr] and [RFC8801] to resolve names within dnsZones.

Figure 2: Verifying Claims using Public Resolver

7.1.2. Verification using DNSSEC

The figure below shows the steps performed to verify the local claims of DNS authority using DNSSEC.

Steps 1-2: The DNSSEC-validating client queries the network encrypted resolver to issue NS queries for the domains in dnsZones. The NS lookup for "example.com" will return a signed response containing "ns1.example.com" and "ns2.example.com". The client then performs full DNSSEC validation locally.

Step 3: As the DNSSEC validation is successful and the network-provided nameservers are the same as the names in the DNSSEC response, and the network-designated resolver’s certificate includes at least one of the names returned in the DNSSEC response, the client has finished validation that the nameservers signaled in [I-D.ietf-add-dnr] and [RFC8801] are owned and managed by the same entity that published the NS records on the Internet. The endpoint will then use that information from [I-D.ietf-add-dnr] and [RFC8801] to resolve names within dnsZones.

```
+---------+                                                +--------------------+
<p>| client  |                                                | Network encrypted resolver |
|---------|                                                |---------------------------|
| DNSSEC OK (DO), NS? example.com (1)                   |                            |</p>
<table>
<thead>
<tr>
<th>NS=ns1.example.com,ns2.example.com, Signed Answer (RRSIG) (2)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- DNSKEY+NS matches RRSIG, use NS</td>
<td></td>
</tr>
<tr>
<td>- both DNR ADNs are authorized</td>
<td></td>
</tr>
<tr>
<td>- finished validation</td>
<td></td>
</tr>
<tr>
<td>use encrypted network-designated resolver for example.com (3)</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 3: Verifying Claims using DNSSEC
7.2. Split-Horizon Only Subdomain of Zone

A subdomain can also be used for all internal DNS names (e.g., the zone internal.example.com exists only on the internal DNS server). For successful validation described in this document the internal DNS server will need a certificate signed by a CA trusted by the client.

For such a name internal.example.com the message flow is similar to Section 7.1 the difference is that queries for hosts not within the subdomain (www.example.com) are sent to the public resolver rather than resolver for internal.example.com.

8. Validation with IKEv2

When the VPN tunnel is IPsec, the encrypted DNS resolver hosted by the VPN service provider can be securely discovered by the endpoint using the ENCDNS_IP_* IKEv2 Configuration Payload Attribute Types defined in [I-D.ietf-ipsecme-add-ike].

Other VPN tunnel types have similar configuration capabilities, not detailed here.

9. Security Considerations

This specification does not alter DNSSEC validation behaviour. To ensure compatibility with validating clients, network operators MUST ensure that names under the split-horizon are correctly signed or place them in an unsigned zone.

If an internal zone name (e.g., internal.example.com) is used with in conjunction with this specification and a public certificate is obtained for validation, that internal zone name will exist in Certificate Transparency [RFC9162] logs. It should be noted, however, that this specification does not leak individual host names (e.g., www.internal.example.com) into the Certificate Transparency logs or to public DNS resolvers.

10. IANA Considerations

This document has no IANA actions.

11. Acknowledgements

Thanks to Mohamed Boucadair, Jim Reid, Tommy Pauly, Paul Vixie, Paul Wouters and Vinny Parla for the discussion and comments.

12. References
12.1. Normative References


12.2. Informative References


Authors’ Addresses

Tirumaleswar Reddy
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
United States of America
Email: danwing@gmail.com

Kevin Smith
Vodafone Group
One Kingdom Street
London
United Kingdom
Email: kevin.smith@vodafone.com

DNS Resolver Information
draft-reddy-add-resolver-info-05

Abstract

This document specifies a method for DNS resolvers to publish information about themselves. Clients can use the resolver information to identify the capabilities of DNS resolvers.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

Historically, DNS stub resolvers communicated with recursive resolvers without needing to know anything about the features supported by these recursive resolvers. As more and more recursive resolvers expose different features that may impact the delivered DNS service, means to help stub resolvers to identify the capabilities of the resolver are valuable. Typically, stub resolvers can discover and authenticate encrypted DNS servers provided by a local network, for example, using the techniques specified in [I-D.ietf-add-dnr] and [I-D.ietf-add-ddr]. However, these stub resolvers need a means to retrieve information from the discovered recursive resolvers about their capabilities.

This document fills that void by specifying a method for stub resolvers to retrieve such information. To that aim, a new RRtype is defined for stub resolvers to query the recursive resolvers. The information that a resolver might want to give is defined in Section 5.

Retrieved information can be used to feed the server selection procedure.

2. Terminology

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.
This document makes use of the terms defined in [RFC8499].

'Encrypted DNS' refers to a DNS protocol that provides an encrypted channel between a DNS client and server (e.g., DoT, DoH, or DoQ).

3. Retrieving Resolver Information

A stub resolver that wants to retrieve the resolver information may use the RRtype "RESINFO" defined in this document (see Section 7.1).

The content of the RDATA in a response to RRtype query is defined in Section 5. If the resolver understands the RESINFO RRtype, the RRset in the Answer section MUST have exactly one record.

The client can retrieve the resolver information using the RESINFO RRtype and QNAME of the domain name that is used to authenticate the DNS server (referred to as ADN in [I-D.ietf-add-dnr]).

If the special use domain name "resolver.arpa" defined in [I-D.ietf-add-ddr] is used to discover the Encrypted DNS server, the client can retrieve the resolver information using the RESINFO RRtype and QNAME of the designated resolver.

4. Format of the Resolver Information

The resolver information is returned as a JSON object. Precisely, the JSON object MUST use the I-JSON message format [RFC7493].

Note that [RFC7493] was based on [RFC7159], but [RFC7159] was replaced by [RFC8259]. Requiring the use of I-JSON instead of more general JSON format greatly increases the likelihood of interoperability.

The JSON object returned by a DNS query may contain any name/value pairs. All names MUST consist only of lower-case ASCII characters, digits, and hyphens (that is, Unicode characters U+0061 through 007A, U+0030 through U+0039, and U+002D). These names MUST be 63 characters or shorter.

All names in the returned object MUST either be defined in the IANA registry Section 7.2 or begin with the substring "temp-" for names defined for local use only.

5. Resolver Information

The resolver information includes the following attributes:

qnameminimization: If the DNS server supports QNAME minimisation
[RFC7816] to improve DNS privacy, the parameter value is set to true. This is a mandatory attribute.

extendeddnsserror: If the DNS server supports extended DNS error (EDE) [RFC8914] to return additional information about the cause of DNS errors, the parameter lists the possible extended DNS error codes that can be returned by the DNS server. This is an optional attribute.

resinfourl: An URL that points to the generic unstructured resolver information (e.g., DoH APIs supported, possible HTTP status codes returned by the DoH server, how to report a problem) for troubleshooting purpose. The server MUST support the content-type 'text/html'. The DNS client MUST reject the URL if the scheme is not "https". The client MUST validate that both the encrypted DNS server and the resolver information server are owned and managed by the same entity by establishing a TLS connection to the domain name in the URL and checking if the subjectAltName entry in the server certificate includes the name of the encrypted DNS server. If this match fails, the client MUST ignore the resolver information. As such, the URL should be treated only as diagnostic information for IT staff. This is a mandatory attribute.

New attributes can be defined as per the procedure defined in Section 7.2.

As specified in [RFC7493], the I-JSON object is encoded as UTF8. [RFC7493] explicitly allows the returned objects to be in any order.

Figure 1 shows an example of resolver information.

```
{
    "qnameminimization": true,
    "extendeddnsserror": [
        15,
        16,
        17
    ],
    "resinfourl": "https://resolver.example.com/guide",
}
```

Figure 1: An Example of Resolver Information
6. Security Considerations

Unless a DNS request to retrieve the resolver information is encrypted (e.g., sent over DNS-over-TLS (DoT) [RFC7858] or DNS-over-HTTPS (DoH)) [RFC8484], the response is susceptible to forgery. The DNS resolver information can be retrieved after the encrypted connection is established to the DNS server or retrieved before the encrypted connection is established to the DNS server by using local DNSSEC validation.

7. IANA Considerations

Note to the RFC Editor: Please update [RFCXXXX] with the RFC number to be assigned to this document.

7.1. RESINFO RRtype

This document requests IANA to register a new value from the "Resource Record (RR) TYPES" subregistry of the "Domain Name System (DNS) Parameters" registry available at [RRTYPE]:

Type: RESINFO
Value: TBD
Meaning: Resolver Information as an I-JSON
Reference: [RFCXXXX]

7.2. DNS Resolver Information Registration

This document requests IANA to create a new registry entitled "DNS Resolver Information". This registry contains definitions of the names that can be used to provide the resolver information.

The registration procedure is Specification Required (Section 4.6 of [RFC8126]).

The structure of the registry is as follows:

Name: The name to be used in the JSON object. The name MUST conform to the definition of "string" in I-JSON message format. The IANA registry MUST NOT register names that begin with "temp-", so these names can be used freely by any implementer.

Value Type: The type of data to be used in the JSON object.

Description: Provides a description of the attribute

Specification: The reference specification for the registered element.
The initial content of this registry is provided in Table 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value Type</th>
<th>Specification</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>qnameminimization</td>
<td>boolean</td>
<td>Indicates whether qnameminimization is enabled or not</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>extendeddnserror</td>
<td>number</td>
<td>Lists the set of extended DNS errors</td>
<td>[RFCXXXX]</td>
</tr>
<tr>
<td>resinfourl</td>
<td>string</td>
<td>Provides an unstructured resolver information that is used for troubleshooting</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

Table 1: Initial RESINFO Registry

8. Acknowledgments

This specification leverages the work that has been documented in [I-D.pp-add-resinfo].

Thanks to Tommy Jensen, Vittorio Bertola, Vinny Parla, Chris Box, Ben Schwartz, Tony Finch, Daniel Kahn Gillmor, Eric Rescorla and Shashank Jain for the discussion and comments.

9. References

9.1. Normative References


9.2. Informative References

[I-D.ietf-add-ddr]

[I-D.ietf-add-dnr]

[I-D.pp-add-resinfo]


Authors’ Addresses

Tirumaleswar Reddy
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India
Email: kondtir@gmail.com

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com
Abstract

This draft describes how the Discovery of Designated Resolvers (DDR) standard interacts with legacy DNS forwarders, including potential incompatibilities and relevant mitigations.

Discussion Venues

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

Discussion of this document takes place on the mailing list (add@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/add/.

Source for this draft and an issue tracker can be found at https://github.com/bemasc/ddr-forwarders.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Conventions and Definitions

Private IP Address - Any IP address reserved for loopback [RFC1122], link-local [RFC3927], private [RFC1918], local [RFC4193], or Carrier-Grade NAT [RFC6598] use.
Legacy DNS Forwarder - An apparent DNS resolver, known to the client only by a private IP address, that forwards the client’s queries to an upstream resolver, and has not been updated with any knowledge of DDR.

Cross-Forwarder Upgrade - Establishment of a direct, encrypted connection between the client and the upstream resolver.

2. Introduction

2.1. Background

The Discovery of Designated Resolvers specification [DDR] describes a mechanism for clients to learn about the encrypted protocols supported by a DNS server. It also describes a conservative client validation policy that has strong security properties and is unlikely to create compatibility problems.

On the topic of client validation of encrypted DNS transports, the DDR specification says:

If the IP address of a Designated Resolver differs from that of an Unencrypted Resolver, clients MUST validate that the IP address of the Unencrypted Resolver is covered by the SubjectAlternativeName of the Encrypted Resolver’s TLS certificate

As TLS certificates cannot cover private IP addresses, this prevents clients that are behind a legacy DNS forwarder from connecting directly to the upstream resolver ("cross-forwarder upgrade").

Recent estimates suggest that a large fraction, perhaps a majority, of residential internet users in the United States and Europe rely on local DNS forwarders that are not compatible with DDR.

2.2. Scope

This informational document describes the interaction between DDR and legacy DNS forwarders. It discusses possible client policies, problems that might arise, and relevant mitigations.

DNS forwarders and resolvers that are implemented with awareness of DDR are out of scope, as they are not affected by this discussion (although see Security Considerations, Section 6).

IPv6-only networks whose default DNS server has a Global Unicast Address are out of scope, even if this server is actually a simple forwarder. If the DNS server does not use a private IP address, it is not a "legacy DNS forwarder" under this draft’s definition.
3. Relaxed Validation client policy

We define a "relaxed validation" client policy as a client behavior that removes the certificate validation requirement when the Unencrypted Resolver is identified by a private IP address, regardless of the Designated Resolver’s IP address. Instead, under this condition, the client connects using the Opportunistic Privacy Profile of encrypted DNS ([RFC7858], Section 4.1).

The Opportunistic Privacy Profile is a broad category, including clients that "might or might not validate" the TLS certificate chain even though there is no authentication identity for the server. This kind of validation can be valuable when combined with a reputation system or a user approval step (see Section 6.1.3 and Section 7.4).

This client policy is otherwise identical to the one described in Section 4 of [DDR].

4. Naturally compatible behaviors

The following system behaviors are naturally compatible with relaxed validation.

4.1. Compatible behaviors in the local network

4.1.1. Malware and threat domain filtering

Certain DNS forwarders block access to domains associated with malware and other threats. Such threats rely on frequently changing domains, so these forwarders necessarily maintain an actively curated list of domains to block. To ensure that this service is not lost due to a cross-forwarder upgrade, the maintainers can simply add "resolver.arpa" to the list.

This pattern has been deployed by Mozilla, with the domain "use-application-dns.net" [MOZILLA-CANARY].

4.1.2. Service category restrictions

Certain DNS forwarders may block access to domains based on the category of service provided by those domains, e.g. domains hosting services that are not appropriate for a work or school environment. As in the previous section, this requires an actively curated list of domains, because the set of domains that offer a given type of service is constantly changing. An actively managed blocking list can easily be revised to include "resolver.arpa".
4.1.3. Time of use restrictions

Certain networks may impose restrictions on the time or duration of use by certain users. This behavior is necessarily implemented below the DNS layer, because DNS-based blocking would be ineffective due to stub resolver caching, so it is not affected by changes in the DNS resolver.

4.2. Upstream resolver services

The forwarder’s upstream resolver might provide additional services, such as filtering. These services are generally independent of cross-forwarder upgrade, and hence naturally compatible.

In special cases where the upstream resolver requires cooperation from a legacy forwarder (e.g. for marking certain queries), one solution is for the upstream resolver to choose not to deploy DDR until all cooperating forwarders have been upgraded. Alternatively, each legacy forwarder can block "resolver.arpa" as described above.

5. Privacy Considerations

The conservative validation policy results in no encryption when a legacy DNS forwarder is present. This leaves the user’s query activity vulnerable to passive monitoring [RFC7258], either on the local network or between the user and the upstream resolver.

The relaxed validation policy allows the use of encrypted transport in these configurations, reducing exposure to a passive surveillance adversary.

6. Security Considerations

When the client uses the conservative validation policy described in [DDR], and a DDR-enabled resolver is identified by a private IP address, the client can establish a secure DDR connection only in the absence of an active attacker. An on-path attacker can impersonate the resolver and intercept all queries, by preventing the DDR upgrade or advertising their own DDR endpoint.

These basic security properties also apply if the client uses the relaxed validation policy described in Section 3. Nonetheless, there are some subtle but important differences in the security properties of these two policies.
6.1. Transient attackers

With the conservative validation policy, a transient on-path attacker can only intercept queries for the duration of their active presence on the network, because the client will only send queries to the original (private) server IP address.

With the relaxed validation behavior, a transient on-path attacker could implant a long-lived DDR response in the client’s cache, directing its queries to an attacker-controlled server on the public internet. This would allow the attack to continue long after the attacker has left the network.

Solving or mitigating this attack is of great importance for the user’s security.

6.1.1. Solution: DNR

This attack does not apply if the client and network implement support for Discovery of Network-designated Resolvers, as that mechanism takes precedence over DDR (see Section 3.2 of [DNR]).

6.1.2. Mitigation: Frequent refresh

The client can choose to refresh the DDR record arbitrarily frequently, e.g. by limiting the TTL. For example, by limiting the TTL to 5 minutes, a client could ensure that any attacker can continue to monitor queries for at most 5 minutes after they have left the local network.

6.1.3. Mitigation: Resolver reputation

A relaxed-validation client might choose to accept a potential cross-forwarder upgrade only if the designated encrypted resolver has sufficient reputation, according to some proprietary reputation scheme (e.g. a locally stored list of respectable resolvers). This limits the ability of a DDR forgery attack to cause harm.

Major DoH client implementations already include lists of known resolvers [CHROME-DOH][MICROSOFT-DOH][MOZILLA-TRR].

6.2. Forensic logging

6.2.1. Network-layer logging

With the conservative validation policy, a random sample of IP packets is likely sufficient for manual retrospective detection of an active attack.
With the relaxed validation policy, forensic logs must capture a specific packet (the attacker’s DDR designation response) to enable retrospective detection.

6.2.1.1. Mitigation: Log all DDR responses

Network-layer forensic logs that are not integrated with the resolver can enable detection of these attacks by logging all DDR responses, or more generally all DNS responses. This makes retrospective attack detection straightforward, as the attacker’s DDR response will indicate an unexpected server.

6.2.2. DNS-layer logging

DNS-layer forensic logging conducted by a legacy DNS forwarder would be lost in a cross-forwarder upgrade.

6.2.2.1. Solution: Respond for resolver.arpa

Forwarders that want to observe all queries from relaxed validation clients will have to synthesize their own response for resolver.arpa, either implementing DDR or disabling it.

7. Compatibility Considerations

Using DDR with legacy DNS forwarders also raises several potential concerns related to loss of existing network services.

7.1. Split-horizon namespaces

Some network resolvers contain additional names that are not resolvable in the global DNS. If these local resolvers are also legacy DNS forwarders, a client that performs a cross-forwarder upgrade might lose access to these local names.

7.1.1. Mitigation: NXDOMAIN Fallback

In "NXDOMAIN Fallback", the client repeats a query to the unencrypted resolver if the encrypted resolver returns NXDOMAIN. This allows the resolution of local names, provided they do not collide with globally resolvable names (as required by [RFC2826]).

This is similar to the fallback behavior currently deployed in Mozilla Firefox [FIREFOX-FALLBACK].
NXDOMAIN Fallback results in slight changes to the security and privacy properties of encrypted DNS. Queries for nonexistent names no longer have protection against a local passive adversary, and local names are revealed to the upstream resolver.

NXDOMAIN Fallback is only applicable when a legacy DNS forwarder might be present, i.e. the unencrypted resolver has a private IP address, and the encrypted resolver has a different IP address. In the other DDR configurations, any local names are expected to resolve similarly on both resolvers.

7.2. Interposable domains

An "interposable domain" is a domain whose owner deliberately allows resolvers to forge certain responses. This arrangement is most common for search engines, which often support a configuration where resolvers forge a CNAME record to direct all clients to a child-appropriate instance of the search engine [DUCK-CNAME][BING-CNAME][GOOGLE-CNAME].

Future deployments of interposable domains can instruct administrators to enable or disable DDR when adding the forged record, but forged records in legacy DNS forwarders could be lost due to a cross-forwarder upgrade.

7.2.1. Mitigation: Exemption list

There are a small number of pre-existing interposable domains, largely of interest only to web browsers. Clients can maintain a list of relevant interposable domains and resolve them only via the network’s resolver.

7.3. Caching

Some legacy DNS forwarders also provide a shared cache for all network users. Cross-forwarder upgrades will bypass this cache, resulting in slower DNS resolution.

7.3.1. Mitigation: Stub caches

Clients can compensate partially for any loss of shared caching by implementing local DNS caches. This mitigation is already widely deployed in browsers and operating systems.
7.4. General mitigation: User controls

For these and other compatibility concerns, a possible mitigation is to provide users or administrators with the ability to control whether DDR is used with legacy forwarders. For example, this control could be provided via a general preference, or via a notification upon discovering a new upstream resolver.

8. Informative References

[ Bing-CNAME ]

[ Chrome-DOH ]

[ DDR ]

[ DNR ]

[ DUCK-CNAME ]

[ Firefox-Fallback ]

[ Google-CNAME ]
Acknowledgments

Thanks to Anthony Lieuallen and Eric Orth for early reviews.

Authors’ Addresses

Benjamin Schwartz
Google LLC

Email: bemasc@google.com

Chris Box
BT

Email: chris.box@bt.com