

dprive
Internet-Draft
Updates: 1035, 7230 (if approved)
Intended status: Informational
Expires: November 18, 2017

D. Gillmor
ACLU
May 17, 2017

Demultiplexing Streamed DNS from HTTP/1.x
draft-dkg-dprive-demux-dns-http-03

Abstract

DNS over TCP and HTTP/1.x are both stream-oriented, client-speaks-first protocols. They can both be run over a stream-based security protocol like TLS. A server accepting a stream-based client can distinguish between a valid stream of DNS queries and valid stream of HTTP/1.x requests by simple observation of the first few octets sent by the client. This can be done without any external demultiplexing mechanism like TCP port number or ALPN.

Implicit multiplexing of the two protocols over a single listening port can be useful for obscuring the presence of DNS queries from a network observer, which makes it relevant for DNS privacy.

Widespread adoption of the described approach could constrain evolution of the stream-based variants of both DNS ([RFC1035]) and HTTP/1.x ([RFC7230]) by ossifying existing distinguishing bit patterns in early octets sent by the client. However, this draft explicitly rules out multiplexing in this form with HTTP/2, so it should place no constraints on it or any higher version of HTTP.

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 <http://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 November 18, 2017.

Copyright Notice

Copyright (c) 2017 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 (<http://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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. Scoping	4
2.1. Distinguish only at the start of a stream	4
2.2. HTTP/2 is not always client-speaks-first	4
2.3. Avoid multiplexing in the clear	5
2.4. Avoid mixing with other demultiplexing	5
2.5. Heavily-restricted network environments	5
2.6. Why not ALPN?	5
3. Overview of initial octets	6
3.1. DNS stream initial octets	6
3.2. HTTP/1.x initial octets	7
3.2.1. HTTP/0.9	7
3.2.2. HTTP/1.0 and HTTP/1.1	8
4. Specific octets	9
4.1. octets 0 and 1	9
4.2. octets 2 and 3	9
4.3. octet 4	10
4.4. octet 5	10
4.5. octets 6 and 7	11
4.6. octets 8 through 11	11
4.7. octets 12 and 13	11
5. Combinations of octets	11
5.1. Proof: a valid DNS message cannot be an HTTP/1.x query	12
6. Guidance for Demultiplexing Servers	13
6.1. Without supporting HTTP/0.9	13
6.2. Supporting archaic HTTP/0.9 clients	13
6.3. Signaling demultiplexing capacity	14
7. Guidance for DNS clients	15
7.1. Interpreting failure	16
8. Guidance for HTTP clients	16

9. Security Considerations	16
10. Privacy Considerations	16
11. IANA Considerations	17
12. Document Considerations	17
13. References	17
13.1. Normative References	17
13.2. Informative References	18
Author's Address	19

1. Introduction

DNS and HTTP/1.x are both client-speaks-first protocols capable of running over stream-based transport like TCP, or as the payload of a typical TLS [RFC5246] session.

There are some contexts where it is useful for a server to be able to decide what protocol is used by an incoming TCP stream, to choose dynamically between DNS and HTTP/1.x on the basis of the stream itself (rather than a port designation or other explicit demultiplexing).

For example, a TLS terminator listening on port 443 and receiving either no ALPN token at all, or the "http/1.1" ALPN token might be willing to serve DNS-over-TLS [RFC7858] as well as HTTPS.

A simple demultiplexing server should do this demuxing based on the first few bytes sent by the client on a given stream; once a choice has been established, the rest of the stream is committed to one or the other interpretation.

This document provides proof that a demultiplexer can robustly distinguish HTTP/1.x from DNS on the basis of the content of the first few bytes of the client's stream alone.

A DNS client that knows it is talking to a server which is this position (e.g. trying to do DNS-over-TLS on TCP port 443 with no ALPN token, used traditionally only for HTTPS) might also want to be aware of network traffic patterns that could confuse such a server. This document presents explicit mitigations that such a DNS client MAY decide to use.

This document limits its discussion to HTTP/1.x over TCP or TLS or some other classical stream-based protocol (it excludes HTTP over QUIC, for example, and HTTP/2 [RFC7540] or later). Likewise, it considers only the TCP variant of DNS (and excludes DNS over UDP or any other datagram transport).

1.1. 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 [RFC2119].

2. Scoping

2.1. Distinguish only at the start of a stream

A server which attempts to distinguish DNS queries from HTTP/1.x requests individually might consider using these guidelines in the middle of a running stream (e.g. at natural boundaries, like the end of an HTTP/1.1 request, or after a DNS message), but this document focuses specifically on a heuristic choice for the whole stream, based on the initial few octets sent by the client.

While it's tempting to consider distinguishing at multiple points in the stream, the complexities of determining the specific end of an HTTP/1.x request body and handling HTTP/1.x error cases make this more difficult to implement on the side of a DNS client configured to talk to such a server. Interleaving the responses themselves on a stream with multiple data elements is also challenging. So do not use this technique anywhere but at the beginning of a stream!

If being able to interleave DNS queries with HTTP requests on a single stream is desired, a strategy like [I-D.hoffman-dns-over-https] or [I-D.ietf-dnsop-dns-wireformat-http] is recommended instead.

2.2. HTTP/2 is not always client-speaks-first

While this demultiplexing technique functions for HTTP/1.0 and HTTP/1.1, it does not work for HTTP/2 [RFC7540] because HTTP/2 is not guaranteed to be a client-speaks-first protocol. In particular, many HTTP/2 servers prefer to send a SETTINGS frame immediately without waiting for data from the client, if they already know they're speaking HTTP/2. In the event that HTTP/2 is to be transported over TLS, the ALPN token negotiated in the TLS handshake is "h2", which allows the server to know as soon as the handshake is complete that it can start pushing data to the client.

A standard DNS-over-TLS client connecting to a server that might be multiplexing DNS with HTTP on the same listener MUST NOT indicate an intent to speak HTTP/2 that could prompt this unsolicited first flight from the server. Concretely, a DNS client connecting over TLS

on TCP port 443 expecting to speak standard DNS-over-TLS [RFC7858] MUST NOT offer or accept the "h2" ALPN token.

If use of DNS in the same channel as HTTP/2 is desired, a strategy like [I-D.hoffman-dns-over-https] is recommended instead.

2.3. Avoid multiplexing in the clear

The widespread deployment of transparent HTTP/1.x proxies makes it likely that any attempt to do this kind of multiplexing/demultiplexing on a cleartext channel that normally carries HTTP/1.x (e.g. TCP port 80) will fail or trigger other "interesting" behaviors. The approach described in this draft should be done only in channels sufficiently obscured that a transparent proxy would not try to interpret the resultant stream.

2.4. Avoid mixing with other demultiplexing

Some other (non-IETF) systems (e.g. [HAPROXY]) take a similar approach with multiplexing data on top of HTTP/1.x by taking advantage of bitpatterns that are presumed to not be present in normal HTTP/1.x requests.

Use of the approach described in this draft in conjunction with these other approaches is not advisable. Doing so safely would require explicit and detailed review of all three (or more) protocols involved.

2.5. Heavily-restricted network environments

Some network environments are so tightly constrained that outbound connections on standard TCP ports are not accessible. In some of these environments, an explicit HTTP proxy is available, and clients must use the HTTP CONNECT pseudo-method to make https connections. While this multiplexing approach can be used in such a restrictive environment, it would be necessary to teach the DNS client how to talk to (and through) the HTTP proxy. These details are out of scope for this document. A DNS client capable of this additional layer of complexity may prefer to pursue a strategy like [I-D.hoffman-dns-over-https] instead.

2.6. Why not ALPN?

If this is done over TLS, a natural question is whether the client should simply indicate its preferred protocol in the TLS handshake's ALPN [RFC7301] extension (e.g. with some new ALPN token "dns").

However, ALPN tokens requested by the client are visible to a network observer (and the ALPN token selected by the server is visible to a network observer in TLS 1.2 and earlier), so a network controller attempting to confine the user's DNS traffic to a limited set of servers could use the ALPN extension as a signal to block DNS-specific streams.

Another alternative could be an ALPN token that indicates potentially-multiplexed traffic (e.g. "http/1.1-or-dns"). This has a comparable problem when confronted with a network adversary that intends to penalize or hamper DNS-over-TLS. Existing HTTP clients will not send this token, and even if some start to offer it, it will provide less cover for DNS-over-TLS clients.

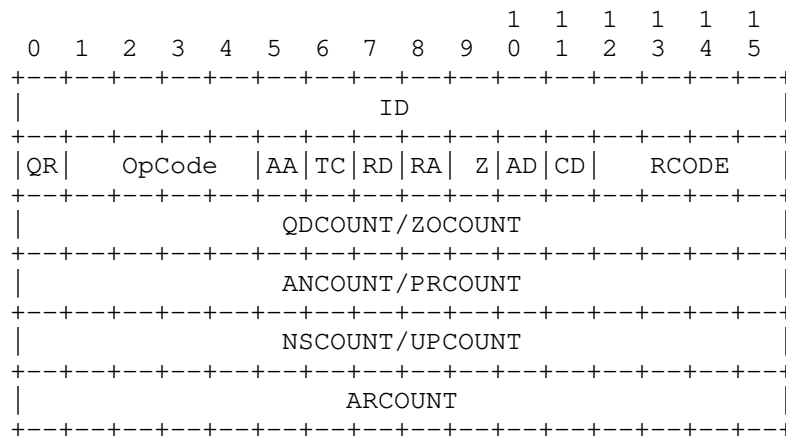
3. Overview of initial octets

3.1. DNS stream initial octets

[RFC1035] section 4.2.2 ("TCP Usage") shows that every stream-based DNS connection starts with a DNS message, preceded with a 2-octet message length field:

The message is prefixed with a two byte length field which gives the message length, excluding the two byte length field.

[RFC6895] section 2 represents the DNS message header section, which is the first part of the DNS message on the wire (after the message length).



So in a DNS over TCP stream, the interpretation of the initial 14 octets are fixed based on information about the first query sent on the stream:

- o 0,1: length of initial DNS message
- o 2,3: DNS Transaction ID
- o 4,5: DNS opcode, flags, and response code
- o 6,7: Question count (or Zone count in UPDATE)
- o 8,9: Answer count (or Prerequisite count in UPDATE)
- o 10,11: Authority count (or Update count in UPDATE)
- o 12,13: Additional RR count

All DNS streams sent over TCP start with at least these 14 octets.

3.2. HTTP/1.x initial octets

In an HTTP stream before HTTP/2, the first octets sent from the client are either the so-called "Simple-Request" (for HTTP/0.9) or the "Request-Line" (for HTTP/1.0 and HTTP/1.1). The data in this initial stream has variable characteristics.

Most servers may wish to ignore the oldest of these, HTTP/0.9.

3.2.1. HTTP/0.9

[RFC1945] section 4.1 says that HTTP/0.9 queries (that is, HTTP queries from before HTTP/1.0 was formalized) use this form:

Simple-Request = "GET" SP Request-URI CRLF

Note that HTTP/0.9 clients send this string and only this string, nothing else (no request body, no subsequent requests). The "Request-URI" token is guaranteed to start with a printable ASCII character, and cannot contain any members of the CTL class (values 0x00 through 0x1F) but due to loose early specifications, it might sometimes contain high-valued octets (those with the most-significant bit set - 0x80 or above).

So the first 5 octets are all constrained to be no less than 0x20 (SP) and no more than 0x7F (DEL), and all subsequent octets sent from the client have a value at least 0x0A (LF).

The shortest possible HTTP/0.9 client request is:

char:	G	E	T	SP	/	CR	LF
index:	0	1	2	3	4	5	6

The lowest possible HTTP/0.9 client request (sorted ASCIIbetically) is:

```
char: G E T SP + : CR LF
index: 0 1 2 3 4 5 6 7
```

3.2.2. HTTP/1.0 and HTTP/1.1

The request line format for HTTP/1.1 matches that of HTTP/1.0 (HTTP/1.1 adds protocol features like pipelining, but doesn't change the request form itself). But unlike HTTP/0.9, the initial verb (the "method") can vary.

[RFC7230] section 3.1.1 says that the first line of an HTTP/1.1 request is:

```
request-line = method SP request-target SP HTTP-version CRLF
method       = token
```

and [RFC7230] section 3.2.6 says:

```
token        = 1*tchar

tchar        = "!" / "#" / "$" / "%" / "&" / "'" / "*"
              / "+" / "-" / "." / "^" / "_" / "`" / "|" / "~"
              / DIGIT / ALPHA
              ; any VCHAR, except delimiters
```

and VCHAR is defined in [RFC5234] appendix B.1 as:

```
VCHAR       = %x21-7E
```

"request-target" itself cannot contain 0x20 (SP) or any CTL characters, or any characters above the US-ASCII range (> 0x7F).

And the "HTTP-version" token is either the literal string "HTTP/1.0" or the literal string "HTTP/1.1", both of which are constrained to the same printable-ASCII range.

The ASCIIbetically-lowest shortest possible HTTP/1.0 or HTTP/1.1 request is:

```
char: ! SP / SP H T T P / 1 . 0 CR LF CR LF
index: 0 1 2 3 4 5 6 7 8 9 0 a b c d e
```

In any case, no HTTP/1.0 or HTTP/1.1 request line can include any values lower than 0x0A (LF) or greater than 0x7F (DEL) in the first 15 octets.

However, [RFC7230] section 3.1.1 also says:

In the interest of robustness, a server that is expecting to receive and parse a request-line SHOULD ignore at least one empty line (CRLF) received prior to the request-line.

So we should also consider accepting an arbitrary number of repeated CRLF sequences before the request-line as a potentially-valid HTTP client behavior.

4. Specific octets

The sections below examine likely values of specific octet positions in the stream. All octet indexes are 0-based.

4.1. octets 0 and 1

Any DNS message less than 3338 octets sent as the initial query over TCP can be reliably distinguished from any version of HTTP/1.x by the first two octets of the TCP stream alone.

3338 is 0x0D0A, or the ASCII string CRLF, which some HTTP/1.x clients might send before an initial request. No HTTP/1.x client can legitimately send anything lower than this.

Most DNS queries are easily within this range automatically.

4.2. octets 2 and 3

In a DNS stream, octets 2 and 3 represent the client-chosen message ID. The message ID is used to bind messages with responses. Over connectionless transports like UDP, this is an important anti-spoofing measure, as well as a distinguishing measure for clients reusing the same UDP port for multiple outstanding queries. Standard DNS clients already explicitly randomize this value.

For the connection-oriented streaming DNS discussed here, the anti-spoofing characteristics are not relevant (the connection itself provides anti-spoofing), so the client is free to choose arbitrary values.

With a standard DNS client which fully-randomizes these values, only 25% of generated queries will have the high bits of both octets set to 0. 100% of all HTTP/1.x requests will have the high bits of both of these octets cleared. Similarly, some small percentage of randomly-generated DNS queries will have values here lower than 0x0A, while no HTTP/1.x clients will ever send these low values.

4.3. octet 4

In a DNS stream, octet 4 combines several fields:

```

    0  1  2  3  4  5  6  7
+---+---+---+---+---+---+---+
|QR|   Opcode   |AA|TC|RD|
+---+---+---+---+---+---+---+

```

In a standard DNS query sent over a streaming interface, QR, Opcode, AA, and TC are all set to 0. The least-significant bit (RD - Recursion Desired) is set when a packet is sent from a stub to a recursive resolver. The value of such an octet is 0x01. This value never occurs in octet 4 of a legitimate HTTP/1.x client.

But under DNS UPDATE ([RFC2136], Opcode is set to 5 and all the option bits are cleared, which means this value would have 0x40 (ASCII '@'), which could legitimately occur in some HTTP/1.x requests at this position.

4.4. octet 5

In a DNS stream, octet 5 also combines several fields:

```

    0  1  2  3  4  5  6  7
+---+---+---+---+---+---+---+
|RA| Z|AD|CD|   RCODE   |
+---+---+---+---+---+---+---+

```

In some DNS messages sent from a client, all these bits are 0. However, section 5.7 of [RFC6840] suggests that queries may wish to set the AD bit to indicate a desire to learn from a validating resolver whether the resolver considers the contents to be Authentic Data.

[RFC6840] also suggests that:

validating resolvers SHOULD set the CD bit on every upstream query.

So many queries, particularly from DNSSEC-validating DNS clients, are likely to set bits 2 and 3, resulting in a value 0x30 (ASCII '0'). This is usually a legitimate value for octet 5 in an HTTP/1.x request.

4.5. octets 6 and 7

In DNS, octets 6 and 7 represent the query count. Most DNS clients will send one query at a time, which makes this value 0x0001. As long as the number of initial queries does not exceed 0x0A0A (2570), then at least one of these octets will have a value less than 0x0A. No HTTP/1.x client sends an octet less than 0x0A in positions 6 or 7.

In DNS UPDATE, octets 6 and 7 represent the zone count. Entries in the Zone section of the DNS UPDATE message are structured identically to entries in the Query section of a standard DNS message.

4.6. octets 8 through 11

In streaming DNS, octets 8 through 11 represent answer counts and authority counts in normal DNS queries, or Prerequisite and Update counts in DNS UPDATE. Standard DNS queries will set them both 0. DNS UPDATE queries are likely to include some records in these sections, so they won't be all zero, but as long as no more than 2570 Prerequisite records and no more than 2570 Update records are sent, at least one octet will have value less than 0x0A. But no HTTP/1.x client sends an octet less than 0x0A in these positions.

4.7. octets 12 and 13

In streaming DNS, octets 12 and 13 represent the number of Additional RRs. When a DNS query is sent with EDNS(0), the OPT RR is accounted for here. So this is often either 0x0000 or 0x0001. In a Secure DNS UPDATE [RFC3007], the SIG(0) or TSIG record is also found in this section, which could increase the values of these octets to 0x0002. No HTTP/1.x client will send octets with these low values at these positions.

5. Combinations of octets

In a DNS message, each Question in the Question section (or Zone in the Zone section for DNS UPDATE) is at least 5 octets (1 octet for zero-length QNAME + 2 octets for QTYPE + 2 octets for QCLASS), and each RR (in the Answer, Authority, and Additional sections for normal DNS queries; or in the Prerequisite, Update, and Additional sections for DNS UPDATE) is at least 11 octets. And the header itself is 12 octets.

So we know that for a valid DNS stream, the first message has a size of at least:

$$\begin{aligned} \text{min_first_msg_size} = & 12 + 5 * (256 * o[6] + o[7]) + \\ & 11 * (256 * (o[8] + o[10] + o[12]) + \\ & \quad o[9] + o[11] + o[13]) \end{aligned}$$

It's possible to compare this value with the expected first query size:

$$\text{first_msg_size} = 256 * o[0] + o[1]$$

if "first_query_size" is less than "min_first_query_size" we can be confident that the stream is not DNS.

5.1. Proof: a valid DNS message cannot be an HTTP/1.x query

For any a valid, stream-based DNS message:

- o If there are fewer than 0x0A00 Questions then octet 6 < 0x0A.
- o If there are fewer than 0x0A00 Answer RRs, then octet 8 < 0x0A.
- o If there are fewer than 0x0A00 Authority RRs, then octet 10 < 0x0A.
- o If there are fewer than 0x0A00 Additional RRs, then octet 12 < 0x0A.

If any of these four inequalities hold, then the packet is clearly DNS, not HTTP/1.x.

if none of them hold, then there are at least 0x0A00 (2560) Questions and $3 * 2560 == 7680$ RRs. But:

$$12 + 5 * 2560 + 11 * 7680 == 97292$$

So the smallest possible DNS message where none of these four inequalities hold is 97292 octets. But a DNS message is limited in size to 65535 octets.

Therefore at least one of these inequalities holds, and one of the first 14 octets of a DNS steam is < 0x0A.

But in a standard HTTP/1.x request, none of the first 14 octets can have a value < 0x0A, so a valid DNS message cannot be mistaken for an HTTP/1.x request.

6. Guidance for Demultiplexing Servers

Upon receiving a connection stream that might be either DNS or HTTP/1.x, a server can inspect the initial octets of the stream to decide where to send it.

6.1. Without supporting HTTP/0.9

A server that doesn't care about HTTP/0.9 can simply wait for the first 14 octets of the client's request to come in. Then the algorithm is:

```
bytestream = read_from_client(14)
for x in bytestream:
    if (x < 0x0A) or (x > 0x7F):
        return 'DNS'
return 'HTTP'
```

6.2. Supporting archaic HTTP/0.9 clients

A server that decides to try to support HTTP/0.9 clients has a slightly more challenging task, since some of them may send fewer octets than the initial DNS message, and the server shouldn't block waiting for data that will never come.

```

bytestream = read_from_client(5)
for x in bytestream[0:5]:
    if (x < 0x0A) or (x > 0x7F):
        return 'DNS'
if (bytestream[0:4] != 'GET '): # not HTTP/0.9
    bytestream += read_from_client(9)
    for x in bytestream[5:14]:
        if (x < 0x0A) or (x > 0x7f):
            return 'DNS'
    return 'HTTP'
else: # maybe HTTP/0.9
    seen_sp = False
    seen_high = False
    while (len(bytestream) < 14):
        if (seen_sp and seen_high):
            return 'DNS'
        x = read_from_client(1)
        bytestream += x
        if (x > 0x7F):
            seen_high = True
        elif (x < 0x0A):
            return 'DNS'
        elif (x == 0x20):
            seen_sp = True # SP found before CRLF, not HTTP/0.9
        elif (x == 0x0A):
            return 'HTTP'
    return 'HTTP'

```

Note that if `read_from_client()` ever fails to read the number of requested bytes (e.g. because of EOF), then the stream is neither valid HTTP nor valid DNS, and can be discarded.

6.3. Signaling demultiplexing capacity

This document assumes that clients can learn out-of-band which listening service they can connect to. For example, the administrator of a machine can configure a local forwarding stub resolver to use DNS-over-TLS on port 443 of some specific server. This explicit configuration carries with it some level of trust - the client is choosing to trust the configured server with its DNS queries.

In some circumstances, it might be useful for a listener to signal to a client that it is willing and capable of handling both DNS and HTTP/1.x traffic. While such signalling could be useful for dynamic discovery, it opens questions of trust (which servers should the client be willing to rely on for DNS resolution?) and is out-of-scope for this draft.

7. Guidance for DNS clients

Consider a DNS client that connects to a server that might be interested in answering HTTP/1.x requests on the same address/port (or other channel identifier). The client wants to send traffic that is unambiguously DNS traffic to make it easy for the server to distinguish it from inbound HTTP/1.x requests. Fortunately, this is trivial to do. In fact, any sensibly-implemented DNS-over-TLS client can use this approach without modification, just by adjusting the port number of the upstream recursive resolver from 853 to 443.

Such a client should follow these guidelines:

- o Send the DNS message size (a 16-bit integer) together in the same packet with the full header of the first DNS message so that the recipient can review as much as possible of the frame at once. This is a best practice for efficient stream-based DNS anyway.

If the client is concerned about stream fragmentation that it cannot control, and it is talking to a server that might be expecting HTTP/0.9 clients, then the server might not be willing to wait for the full initial 14 octets to make a decision.

Note that this fragmentation is not a concern for streams wrapped in TLS when using modern AEAD ciphersuites. In this case, the client gets to choose the size of the plaintext record, which is either recovered by the server in full (unfragmented) or the connection fails.

If the client does not have such a guarantee from the transport, it MAY also take one of the following mitigating actions relating to the first DNS message it sends in the stream [explanation of what the server gets to see in the fragmented stream case are in square brackets after each mitigation]:

- o Ensure the first message is marked as a query (QR = 0), and it uses opcode 0 ("Standard Query"). [bytestream[4] < 0x08]
- o Ensure that the first message has RA = 0, Z = 0, and RCODE = 0. [bytestream[5] == 0x00]
- o Ensure that the high bit of the first octet of the message ID of the first message is set. [bytestream[2] > 0x7F]
- o Send an initial short Server Status DNS message ahead of the otherwise intended initial DNS message. [bytestream[0] == 0x00]

- o Use the EDNS(0) padding option [RFC7830] to pad the first message to a multiple of 256 octets. [bytestream[1] == 0x00]

7.1. Interpreting failure

FIXME: A DNS client that does not already know that a server is willing to carry both types of traffic SHOULD expect a transport connection failure of some sort. Can we say something specific about what it should expect?

8. Guidance for HTTP clients

HTTP clients SHOULD NOT send HTTP/0.9 requests, since modern HTTP servers are not required to support HTTP/0.9. Sending an HTTP/1.0 request (or any later version) is sufficient for a server to be able to distinguish the two protocols.

9. Security Considerations

FIXME: Clients should locally validate DNSSEC (servers may still be able to omit some records)

FIXME: if widely deployed, consider amplification for DDoS against authoritative servers?

FIXME: consider DNSSEC transparency

FIXME: consider TLS session resumption - this counts as a new stream boundary, so the multiplexing decision need not persist across resumption.

FIXME: consider 0-RTT

FIXME: consider X.509 cert validation

FIXME: what other security considerations should clients take?

FIXME: what other security considerations should servers take?

10. Privacy Considerations

FIXME: DNS queries and HTTP requests can reveal potentially sensitive information about the sender.

FIXME: consider DNS and HTTP traffic analysis - how should requests or responses be padded, aggregated, or delayed given that streams are multiplexed?

FIXME: any other privacy considerations?

11. IANA Considerations

This document does not ask IANA to make any changes to existing registries.

However, it does update the DNS and HTTP specifications, to reflect the fact that services using this demultiplexing technique may be constrained in adoption of future versions of either stream-based DNS or HTTP/1.x if those future versions modify either protocol in a way that breaks with the distinctions documented here.

In particular, this draft assumes that all future stream-based versions of HTTP/1.x should have the following properties:

- o the client will speak first
- o the client will send at least 14 octets before expecting a response from the server.
- o none of those first 14 octets will be below 0x0A (LF) or above 0x7F (DEL).

Future extensions to stream-based DNS or HTTP/1.x should take this demultiplexing technique into consideration.

12. 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/hddemux> or by e-mail to the author. Please direct all significant commentary to the public IETF DNS Privacy mailing list: dns-privacy@ietf.org or to the IETF HTTP WG mailing list: ietf-http-wg@w3.org

13. References

13.1. Normative References

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

- [RFC1945] Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, DOI 10.17487/RFC1945, May 1996, <<http://www.rfc-editor.org/info/rfc1945>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<http://www.rfc-editor.org/info/rfc5234>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.

13.2. Informative References

- [HAPROXY] Tarreau, W., "The Proxy protocol", March 2017, <<https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt>>.
- [I-D.hoffman-dns-over-https] Hoffman, P. and P. McManus, "DNS Queries over HTTPS", draft-hoffman-dns-over-https-00 (work in progress), May 2017.
- [I-D.ietf-dnsop-dns-wireformat-http] Song, L., Vixie, P., Kerr, S., and R. Wan, "DNS wire-format over HTTP", draft-ietf-dnsop-dns-wireformat-http-01 (work in progress), March 2017.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<http://www.rfc-editor.org/info/rfc3007>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <<http://www.rfc-editor.org/info/rfc6840>>.
- [RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <<http://www.rfc-editor.org/info/rfc6895>>.
- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<http://www.rfc-editor.org/info/rfc7301>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<http://www.rfc-editor.org/info/rfc7540>>.
- [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <<http://www.rfc-editor.org/info/rfc7830>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<http://www.rfc-editor.org/info/rfc7858>>.

Author's Address

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY 10004
USA

Email: dkg@fifthhorseman.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 10, 2020

C. Huitema
Private Octopus Inc.
M. Shore
Fastly
A. Mankin
Salesforce
S. Dickinson
Sinodun IT
J. Iyengar
Fastly
September 7, 2019

Specification of DNS over Dedicated QUIC Connections
draft-huitema-quic-dnsquic-07

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 10, 2020.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Key Words	4
3. Design Considerations	4
3.1. Scope is Limited to the Stub to Resolver Scenario	4
3.2. Provide DNS Privacy	5
3.3. Design for Minimum Latency	5
3.4. Development of QUIC Protocols and API	6
3.5. No Specific Middlebox Bypass Mechanism	6
4. Specifications	7
4.1. Connection Establishment	7
4.1.1. Draft Version Identification	7
4.1.2. Port Selection	7
4.2. Stream Mapping and Usage	8
4.2.1. Server Initiated Transactions	8
4.2.2. Stream Reset	9
4.3. Closing the DNS/QUIC Connection	9
4.4. Connection Resume and 0-RTT	9
5. Implementation Requirements	10
5.1. Authentication	10
5.2. Fall Back to Other Protocols on Connection Failure	10
5.3. Response Sizes	10
5.4. DNS Message IDs	10
5.5. Padding	11
5.6. Connection Handling	11
5.6.1. Connection Reuse	11
5.6.2. Connection Close	11
5.6.3. Idle Timeouts	12
5.7. Flow Control Mechanisms	12
6. Security Considerations	12
7. Privacy Considerations	13
7.1. Privacy Issues With Zero RTT data	13
7.2. Privacy Issues With Session Resume	13
7.3. Traffic Analysis	14
8. IANA Considerations	14
8.1. Registration of DNS/QUIC Identification String	14
8.2. Reservation of Dedicated Port	15

8.2.1. Port number 784 for experimentations	15
9. Acknowledgements	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Authors' Addresses	18

1. Introduction

Domain Name System (DNS) concepts are specified in [RFC1034]. This document presents a mapping of the DNS protocol [RFC1035] over QUIC transport [I-D.ietf-quic-transport] [I-D.ietf-quic-tls]. The goals of this mapping are:

1. Provide the same DNS privacy protection as DNS over TLS (DNS/TLS) [RFC7858]. This includes an option for the client to authenticate the server by means of an authentication domain name [RFC8310].
2. Provide an improved level of source address validation for DNS servers compared to DNS/UDP [RFC1035].
3. Provide a transport that is not constrained by path MTU limitations on the size of DNS responses it can send.
4. Explore the potential performance gains of using QUIC as a DNS transport, versus other solutions like DNS over UDP (DNS/UDP) [RFC1035] or DNS/TLS [RFC7858].
5. Participate in the definition of QUIC protocols and API, by outlining a use case for QUIC different from HTTP over QUIC [I-D.ietf-quic-http].

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

1. No attempt is made to support zone transfers [RFC5936], as these are not relevant to the stub to recursive resolver scenario.
2. No attempt is made to evade potential blocking of DNS/QUIC traffic by middleboxes.

Users interested in zone transfers should continue using TCP based solutions. Users interested in evading middleboxes should consider using solutions like DNS/HTTPS [RFC8484].

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

In this document, Section 3 presents the reasoning that guided the proposed design. Section 4 specifies the actual mapping of DNS/QUIC. Section 5 presents guidelines on the implementation, usage and deployment of DNS/QUIC.

2. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC8174].

3. Design Considerations

This section and its subsection present the design guidelines that were used for the proposed mapping of DNS/QUIC. This section is informative in nature.

3.1. Scope is Limited to the Stub to Resolver Scenario

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

QUESTION: Should this document specify any aspects of configuration of discoverability differently to DNS/TLS?

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

The DNS protocol is also used for zone transfers. In the zone transfer scenario ([RFC5936]), the client emits a single AXFR query,

and the server responds with a series of AXFR responses. This creates a unique profile, in which a query results in several responses. Supporting that profile would complicate the mapping of DNS queries over QUIC streams. Zone transfers are not used in the stub to recursive scenario that is the focus here, and seem to be currently well served by the DNS over TCP (DNS/TCP). There is no attempt to support them in this proposed mapping of DNS to QUIC.

3.2. Provide DNS Privacy

DNS privacy considerations are described in [RFC7626]. [RFC7858] defines how to mitigate some of these issues by transmitting DNS messages over TLS and TCP and [RFC8310] specifies Strict and Opportunistic Usage Profiles for DNS/TLS including how stub resolvers can authenticate recursive resolvers.

QUIC connection setup includes the negotiation of security parameters using TLS, as specified in [I-D.ietf-quic-tls], enabling encryption of the QUIC transport. Transmitting DNS messages over QUIC will provide essentially the same privacy protections as [RFC7858] and [RFC8310]. Further discussion on this is provided in Section 7.

3.3. Design for Minimum Latency

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

1. Support for 0-RTT data during session resumption.
2. Support for advanced error recovery procedures as specified in [I-D.ietf-quic-recovery].
3. Mitigation of head-of-line blocking by allowing parallel delivery of data on multiple streams.

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

1. Optional support for sending 0-RTT data during session resumption (the security and privacy implications of this are discussed in later sections).
2. Long-lived QUIC connections over which multiple DNS transactions are performed, generating the sustained traffic required to benefit from advanced recovery features.

3. Mapping of each DNS Query/Response transaction to a separate stream, to mitigate head-of-line blocking.

These considerations will be reflected in the mapping of DNS traffic to QUIC streams in Section 4.2.

3.4. Development of QUIC Protocols and API

QUIC is defined as a layered protocol, with application-specific mapping layered on top of the generic QUIC transport. The only mapping defined at this stage is HTTP over QUIC [I-D.ietf-quic-http]. Adding a different mapping for a different application contributes to the development of QUIC.

HTTP/QUIC parallels the definition of HTTP/2.0, in which HTTP queries and responses are carried as series of frames. The HTTP/QUIC mapping provide with some simplification compared to HTTP/TLS/TCP, as QUIC already provides concepts like stream identification or end of stream marks. Dedicated control channel are used to carry connection data, such as settings or the relative priority of queries. It would be completely possible to use the HTTP/QUIC mapping to carry DNS requests as HTTP queries, as specified in [RFC8484]. We are somewhat concerned that this mapping carries the overhead of HTTP into the DNS protocol, resulting in additional complexity and overhead.

In this document a different design is deliberately explored, in which clients and servers can initiate queries as determined by the DNS application logic, opening new streams as necessary. This provides for maximum parallelism between queries, as noted in Section 3.3. It also places constraints on the API. Client and servers will have to be notified of the opening of a new stream by their peer. Instead of orderly closing the control stream, client and server will have to use orderly connection closure mechanisms and manage the potential loss of data if closing on one end conflicts with opening of a stream on the other end.

3.5. No Specific Middlebox Bypass Mechanism

Being different from HTTP/QUIC is a design choice. The advantage is that the mapping can be defined for minimal overhead and maximum performance. The downside is that the difference can be noted by firewalls and middleboxes. There may be environments in which HTTP/QUIC will be allowed, but DNS/QUIC will be disallowed and blocked by these middle boxes.

It is recognized that this might be a problem, but there is currently no indication on how widespread that problem might be. The problem might be acute enough that the only realistic solution would be to

communicate with independent recursive resolvers using DNS/HTTPS, or maybe DNS/HTTP/QUIC. Or the problem might be rare enough and the performance gains significant enough that the appropriate solution would be to use DNS/QUIC most of the time, and fall back to DNS/HTTPS some of the time. Measurements and experimentation will inform that decision.

It may indeed turn out that the complexity and overhead concerns are negligible compared to the potential advantages of DNS/HTTPS, such as integration with web services or firewall traversal, and that DNS/QUIC does not provide sufficient performance gains to justify a new protocol. We will evaluate that once implementations are available and can be compared. In the meanwhile, we believe that a clean design is most likely to inform the QUIC development, as explained in Section 3.4.

4. Specifications

4.1. Connection Establishment

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

4.1.1. Draft Version Identification

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

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

Implementations of draft versions of the protocol **MUST** add the string "-" and the corresponding draft number to the identifier. For example, draft-huitema-quic-dnsoquic-01 is identified using the string "dq-h01".

4.1.2. Port Selection

By default, a DNS server that supports DNS/QUIC **MUST** listen for and accept QUIC connections on the dedicated UDP port TBD (number to be defined in Section 8), unless it has mutual agreement with its clients to use a port other than TBD for DNS/QUIC. In order to use a port other than TBD, both clients and servers would need a configuration option in their software.

By default, a DNS client desiring to use DNS/QUIC with a particular server MUST establish a QUIC connection to UDP port TBD on the server, unless it has mutual agreement with its server to use a port other than port TBD for DNS/QUIC. Such another port MUST NOT be port 53 or port 853. This recommendation against use of port 53 for DNS/QUIC is to avoid confusion between DNS/QUIC and DNS/UDP as specified in [RFC1035]. Similarly, using port 853 would cause confusion between DNS/QUIC and DNS/DTLS as specified in [RFC8094].

4.2. Stream Mapping and Usage

The mapping of DNS traffic over QUIC streams takes advantage of the QUIC stream features detailed in Section 10 of [I-D.ietf-quic-transport].

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

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

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

Therefore, a single client initiated DNS transaction consumes a single stream. This means that the client's first query occurs on QUIC stream 4, the second on 8, and so on.

4.2.1. Server Initiated Transactions

There are planned traffic patterns in which a server sends unsolicited queries to a client, such as for example PUSH messages defined in [I-D.ietf-dnssd-push]. These occur when a client subscribes to changes for a particular DNS RRset or resource record type. When a PUSH server wishes to send such updates it MUST select the next available server initiated bidirectional stream, in conformance with [I-D.ietf-quic-transport].

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

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

Therefore a single server initiated DNS transaction consumes a single stream. This means that the servers's first query occurs on QUIC stream 1, the second on 5, and so on.

4.2.2. Stream Reset

Stream transmission may be abandoned by either party, using the stream "reset" facility. A stream reset indicates that one party is unwilling to continue processing the transaction associated with the stream. The corresponding transaction **MUST** be abandoned. A Server Failure (SERVFAIL, [RFC1035]) **SHOULD** be notified to the initiator of the transaction.

4.3. Closing the DNS/QUIC Connection

QUIC connections are closed using the CONNECTION_CLOSE mechanisms specified in [I-D.ietf-quic-transport]. Connections can be closed at the initiative of either the client or the server (also see Section 5.6.2). The party initiating the connection closure **SHOULD** use the QUIC GOAWAY mechanism to initiate a graceful shutdown of a connection.

The transactions corresponding to stream number higher than indicated in the GO AWAY frames **MUST** be considered failed. Similarly, if streams are still open when the CONNECTION_CLOSE is received, the corresponding transactions **MUST** be considered failed. In both cases, a Server Failure (SERVFAIL, [RFC1035]) **SHOULD** be notified to the initiator of the transaction.

4.4. Connection Resume and 0-RTT

A stub resolver **MAY** take advantage of the connection resume mechanisms supported by QUIC transport [I-D.ietf-quic-transport] and QUIC TLS [I-D.ietf-quic-tls]. Stub resolvers **SHOULD** consider potential privacy issues associated with session resume before deciding to use this mechanism. These privacy issues are detailed in Section 7.2.

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

5. Implementation Requirements

5.1. Authentication

For the stub to recursive resolver scenario, the authentication requirements are the same as described in [RFC7858] and [RFC8310]. There is no need to authenticate the client's identity in either scenario.

5.2. Fall Back to Other Protocols on Connection Failure

If the establishment of the DNS/QUIC connection fails, clients SHOULD attempt to fall back to DNS/TLS and then potentially clear text, as specified in [RFC7858] and [RFC8310], depending on their privacy profile.

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

5.3. Response Sizes

DNS/QUIC does not suffer from the limitation on the size of responses that can be delivered as DNS/UDP [RFC1035] does, since large responses will be sent in separate STREAM frames in separate packets.

QUESTION: However, this raises a new issue because the responses sent over QUIC can be significantly larger than those sent over TCP (65,535 bytes). According to [I-D.ietf-quic-transport] "The largest offset delivered on a stream - the sum of the re-constructed offset and data length - MUST be less than 2^{62} ". Should a specific limit be applied for DNS/QUIC responses or not?

5.4. DNS Message IDs

When sending multiple queries over a QUIC connection, clients MUST NOT reuse the DNS Message ID of an in-flight query on that connection in order to avoid Message ID collisions.

Clients MUST match responses to outstanding queries using the STREAM ID and Message ID and if the response contains a question section, the client MUST match the QNAME, QCLASS, and QTYPE fields. Failure to match is a DNS/QUIC protocol error. Clients observing such errors SHOULD close the connection immediately, indicating the application

specific error code 0x00000001. The client should also mark the server as inappropriate for future use of DNS/QUIC.

5.5. Padding

There are mechanisms specified for both padding individual DNS messages [RFC7830], [RFC8467] and padding within QUIC packets (see Section 8.6 of [I-D.ietf-quic-transport]), which may contain multiple frames.

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

5.6. Connection Handling

[RFC7766] provides updated guidance on DNS/TCP much of which is applicable to DNS/QUIC. This section attempts to specify how those considerations apply to DNS/QUIC.

5.6.1. Connection Reuse

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

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

5.6.2. Connection Close

In order to amortize QUIC and TLS connection setup costs, clients and servers SHOULD NOT immediately close a QUIC connection after each response. Instead, clients and servers SHOULD reuse existing QUIC connections for subsequent queries as long as they have sufficient resources. In some cases, this means that clients and servers may need to keep idle connections open for some amount of time.

Under normal operation DNS clients typically initiate connection closing on idle connections; however, DNS servers can close the connection if the idle timeout set by local policy is exceeded.

Also, connections can be closed by either end under unusual conditions such as defending against an attack or system failure/reboot.

Clients and servers that keep idle connections open MUST be robust to termination of idle connection by either party. As with current DNS over TCP, DNS servers MAY close the connection at any time (perhaps due to resource constraints). As with current DNS/TCP, clients MUST handle abrupt closes and be prepared to reestablish connections and/or retry queries.

5.6.3. Idle Timeouts

Proper management of established and idle connections is important to the healthy operation of a DNS server. An implementation of DNS/QUIC SHOULD follow best practices for DNS/TCP, as described in [RFC7766]. Failure to do so may lead to resource exhaustion and denial of service.

This document does not make specific recommendations for timeout values on idle connections. Clients and servers should reuse and/or close connections depending on the level of available resources. Timeouts may be longer during periods of low activity and shorter during periods of high activity. Current work in this area may also assist DNS/TLS clients and servers in selecting useful timeout values [RFC7828] [RFC8490] [TDNS].

TODO: Clarify what timers (idle timeouts, response timeouts) apply at the stream level and at the connection level.

TODO: QUIC provides an efficient mechanism for resuming connections, including the possibility of sending 0-RTT data. Does that change the tradeoff? Is it plausible to use shorter timers than specified for TCP?

5.7. Flow Control Mechanisms

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

6. Security Considerations

The security considerations of DNS/QUIC should be comparable to those of DNS/TLS [RFC7858].

7. Privacy Considerations

DNS/QUIC is specifically designed to protect the DNS traffic between stub and resolver from observations by third parties, and thus protect the privacy of queries from the stub. However, the recursive resolver has full visibility of the stub's traffic, and could be used as an observation point, as discussed in [RFC7626]. These considerations do not differ between DNS/TLS and DNS/QUIC and are not discussed further here.

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

1. Adversaries could replay the zero-RTT data and infer its content from the behavior of the receiving server.
2. The zero-RTT mechanism relies on TLS resume, which can provide linkability between successive client sessions.

These issues are developed in Section 7.1 and Section 7.2.

7.1. Privacy Issues With Zero RTT data

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

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

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

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

7.2. Privacy Issues With Session Resume

The QUIC session resume mechanism reduces the cost of reestablishing sessions and enables zero-RTT data. There is a linkability issue associated with session resume, if the same resume token is used

several times, but this risk is mitigated by the mechanisms incorporated in QUIC and in TLS 1.3. With these mechanisms, clients and servers can cooperate to avoid linkability by third parties. However, the server will always be able to link the resumed session to the initial session. This creates a virtual long duration session. The series of queries in that session can be used by the server to identify the client.

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

7.3. Traffic Analysis

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

Implementations SHOULD use the mechanisms defined in Section 5.5 to mitigate this attack.

8. IANA Considerations

8.1. Registration of DNS/QUIC Identification String

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

The "dq" string identifies DNS/QUIC:

Protocol: DNS/QUIC

Identification Sequence: 0x64 0x71 ("dq")

Specification: This document

8.2. Reservation of Dedicated Port

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

Service Name	domain-s
Transport Protocol(s)	TCP/UDP
Assignee	IESG
Contact	IETF Chair
Description	DNS query-response protocol run over QUIC
Reference	This document

8.2.1. Port number 784 for experimentations

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

Early experiments MAY use port 784. This port is marked in the IANA registry as unassigned.

9. Acknowledgements

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

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

Thanks to our wide cast of supporters.

10. References

10.1. Normative References

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using TLS to Secure QUIC",
draft-ietf-quic-tls-22 (work in progress), July 2019.

- [I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-22 (work in progress), July 2019.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/info/rfc7301>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.

10.2. Informative References

- [DNSORTT] Kahn Gillmor, D., "DNS + 0-RTT", Message to DNS-Privacy WG mailing list, April 2016, <<https://www.ietf.org/mail-archive/web/dns-privacy/current/msg01276.html>>.
- [I-D.bortzmeyer-dprive-resolver-to-auth]
Bortzmeyer, S., "Encryption and authentication of the DNS resolver-to-authoritative communication", draft-bortzmeyer-dprive-resolver-to-auth-01 (work in progress), March 2018.
- [I-D.ietf-dnssd-push]
Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-24 (work in progress), August 2019.
- [I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-22 (work in progress), July 2019.

- [I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-22 (work in progress), July 2019.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<https://www.rfc-editor.org/info/rfc5936>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <<https://www.rfc-editor.org/info/rfc7120>>.
- [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <<https://www.rfc-editor.org/info/rfc7626>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", RFC 7828, DOI 10.17487/RFC7828, April 2016, <<https://www.rfc-editor.org/info/rfc7828>>.
- [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <<https://www.rfc-editor.org/info/rfc7830>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.

- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <<https://www.rfc-editor.org/info/rfc8467>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <<https://www.rfc-editor.org/info/rfc8490>>.
- [TDNS] Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A., and N. Somaiya, "Connection-Oriented DNS to Improve Privacy and Security", 2015 IEEE Symposium on Security and Privacy (SP), DOI 10.1109/SP.2015.18, <<http://dx.doi.org/10.1109/SP.2015.18>>.

Authors' Addresses

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

Email: huitema@huitema.net

Melinda Shore
Fastly

Email: mshore@fastly.com

Allison Mankin
Salesforce

Email: amankin@salesforce.com

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

Email: sara@sinodun.com

Jana Iyengar
Fastly

Email: jri.ietf@gmail.com

dprive
Internet-Draft
Updates: 7858 (if approved)
Intended status: Standards Track
Expires: March 15, 2018

S. Dickinson
Sinodun
D. Gillmor
ACLU
T. Reddy
McAfee
September 11, 2017

Usage and (D)TLS Profiles for DNS-over-(D)TLS
draft-ietf-dprive-dtls-and-tls-profiles-11

Abstract

This document discusses Usage Profiles, based on one or more authentication mechanisms, which can be used for DNS over Transport Layer Security (TLS) or Datagram TLS (DTLS). These profiles can increase the privacy of DNS transactions compared to using only clear text DNS. This document also specifies new authentication mechanisms - it describes several ways a DNS client can use an authentication domain name to authenticate a (D)TLS connection to a DNS server. Additionally, it defines (D)TLS protocol profiles for DNS clients and servers implementing DNS-over-(D)TLS. This document updates RFC 7858.

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 <http://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 March 15, 2018.

Copyright Notice

Copyright (c) 2017 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 (<http://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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	5
3. Scope	6
4. Discussion	7
5. Usage Profiles	7
5.1. DNS Resolution	10
6. Authentication in DNS-over(D)TLS	10
6.1. DNS-over-(D)TLS Startup Configuration Problems	10
6.2. Credential Verification	11
6.3. Summary of Authentication Mechanisms	11
6.4. Combining Authentication Mechanisms	14
6.5. Authentication in Opportunistic Privacy	14
6.6. Authentication in Strict Privacy	15
6.7. Implementation guidance	15
7. Sources of Authentication Domain Names	15
7.1. Full direct configuration	15
7.2. Direct configuration of ADN only	16
7.3. Dynamic discovery of ADN	16
7.3.1. DHCP	16
8. Authentication Domain Name based Credential Verification	17
8.1. PKIX Certificate Based Authentication	17
8.2. DANE	17
8.2.1. Direct DNS Lookup	18
8.2.2. TLS DNSSEC Chain extension	18
9. (D)TLS Protocol Profile	19
10. IANA Considerations	20
11. Security Considerations	20
11.1. Counter-measures to DNS Traffic Analysis	20
12. Acknowledgments	21
13. References	21
13.1. Normative References	21
13.2. Informative References	23
Appendix A. Server capability probing and caching by DNS clients	24
Appendix B. Changes between revisions	24
B.1. -11 version	25
B.2. -10 version	25

B.3.	-09 version	26
B.4.	-08 version	26
B.5.	-07 version	26
B.6.	-06 version	26
B.7.	-05 version	27
B.8.	-04 version	27
B.9.	-03 version	27
B.10.	-02 version	27
B.11.	-01 version	28
B.12.	draft-ietf-dprive-dtls-and-tls-profiles-00	28
Authors'	Addresses	28

1. Introduction

DNS Privacy issues are discussed in [RFC7626]. The specific issues described there that are most relevant to this document are

- o Passive attacks which eavesdrop on clear text DNS transactions on the wire (Section 2.4) and
- o Active attacks which redirect clients to rogue servers to monitor DNS traffic (Section 2.5.3).

Mitigating against these attacks increases the privacy of DNS transactions, however many of the other issues raised in [RFC7626] still apply.

Two documents that provide ways to increase DNS privacy between DNS clients and DNS servers are:

- o Specification for DNS over Transport Layer Security (TLS) [RFC7858], referred to here as simply 'DNS-over-TLS'
- o DNS over Datagram Transport Layer Security (DTLS) [RFC8094], referred to here simply as 'DNS-over-DTLS'. Note that this document has the Category of Experimental.

Both documents are limited in scope to communications between stub clients and recursive resolvers and the same scope is applied to this document (see Section 2 and Section 3). The proposals here might be adapted or extended in future to be used for recursive clients and authoritative servers, but this application was out of scope for the Working Group charter at the time this document was finished.

This document specifies two Usage Profiles (Strict and Opportunistic) for DTLS [RFC6347] and TLS [RFC5246] which provide improved levels of mitigation against the attacks described above compared to using only clear text DNS.

Section 5 presents a generalized discussion of Usage Profiles by separating the Usage Profile, which is based purely on the security properties it offers the user, from the specific mechanism(s) that are used for DNS server authentication. The Profiles described are:

- o A Strict Profile that requires an encrypted connection and successful authentication of the DNS server which mitigates both passive eavesdropping and client re-direction (at the expense of providing no DNS service if this is not available).
- o An Opportunistic Profile that will attempt, but does not require, encryption and successful authentication; it therefore provides limited or no mitigation against such attacks but offers maximum chance of DNS service.

The above Usage Profiles attempt authentication of the server using at least one authentication mechanism. Section 6.4 discusses how to combine authentication mechanisms to determine the overall authentication result. Depending on that overall authentication result (and whether encryption is available) the Usage Profile will determine if the connection should proceed, fallback or fail.

One authentication mechanism is already described in [RFC7858]. That document specifies a Subject Public Key Info (SPKI) based authentication mechanism for DNS-over-TLS in the context of a specific case of a Strict Usage Profile using that single authentication mechanism. Therefore the "Out-of-band Key-pinned Privacy Profile" described in [RFC7858] would qualify as a "Strict Usage Profile" that used SPKI pinning for authentication.

This document extends the use of SPKI pinset based authentication so that it is considered a general authentication mechanism that can be used with either DNS-over-(D)TLS Usage Profile. That is, the SPKI pinset mechanism described in [RFC7858] MAY be used with DNS-over-(D)TLS.

This document also describes a number of additional authentication mechanisms all of which specify how a DNS client should authenticate a DNS server based on an 'authentication domain name'. In particular, the following is described:

- o How a DNS client can obtain the combination of an authentication domain name and IP address for a DNS server. See Section 7.
- o What are the acceptable credentials a DNS server can present to prove its identity for (D)TLS authentication based on a given authentication domain name. See Section 8.

- o How a DNS client can verify that any given credential matches the authentication domain name obtained for a DNS server. See Section 8.

In Section 9 this document defines a (D)TLS protocol profile for use with DNS. This profile defines the configuration options and protocol extensions required of both parties to optimize connection establishment and session resumption for transporting DNS, and to support all currently specified authentication mechanisms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Several terms are used specifically in the context of this draft:

- o DNS client: a DNS stub resolver or forwarder. In the case of a forwarder, the term "DNS client" is used to discuss the side that sends queries.
- o DNS server: a DNS recursive resolver or forwarder. In the case of a forwarder, the term "DNS server" is used to discuss the side that responds to queries. For emphasis, in this document the term does not apply to authoritative servers.
- o Privacy-enabling DNS server: A DNS server that implements DNS-over-TLS [RFC7858] and may optionally implement DNS-over-DTLS [RFC8094]. The server should also offer at least one of the credentials described in Section 8 and implement the (D)TLS profile described in Section 9.
- o (D)TLS: For brevity this term is used for statements that apply to both Transport Layer Security [RFC5246] and Datagram Transport Layer Security [RFC6347]. Specific terms will be used for any statement that applies to either protocol alone.
- o DNS-over-(D)TLS: For brevity this term is used for statements that apply to both DNS-over-TLS [RFC7858] and DNS-over-DTLS [RFC8094]. Specific terms will be used for any statement that applies to either protocol alone.
- o Authentication domain name: A domain name that can be used to authenticate a privacy-enabling DNS server. Sources of authentication domain names are discussed in Section 7.

- o **SPKI Pinsets:** [RFC7858] describes the use of cryptographic digests to "pin" public key information in a manner similar to HTTP Public Key Pinning [RFC7469] (HPKP). An SPKI pinset is a collection of these pins that constrains a DNS server.
- o **Authentication information:** Information a DNS client may use as the basis of an authentication mechanism. In this context that can be either a:
 - * a SPKI pinset or
 - * an authentication domain name
- o **Reference Identifier:** a Reference Identifier as described in [RFC6125], constructed by the DNS client when performing TLS authentication of a DNS server.
- o **Credential:** Information available for a DNS server which proves its identity for authentication purposes. Credentials discussed here include:
 - * PKIX certificate
 - * DNSSEC validated chain to a TLSA recordbut may also include SPKI pinsets.

3. Scope

This document is limited to describing

- o Usage Profiles based on general authentication mechanisms
- o The details of domain name based authentication of DNS servers by DNS clients (as defined in the terminology section)
- o The (D)TLS profiles needed to support authentication in DNS-over-(D)TLS.

As such, the following things are out of scope:

- o Authentication of authoritative servers by recursive resolvers.
- o Authentication of DNS clients by DNS servers.
- o The details of how to perform SPKI-pinset-based authentication. This is defined in [RFC7858].

- o Any server identifier other than domain names, including IP addresses, organizational names, country of origin, etc.

4. Discussion

One way to mitigate against passive attackers eavesdropping on clear text DNS transactions is to encrypt the query (and response). Such encryption typically provides integrity protection as a side-effect, which means on-path attackers cannot simply inject bogus DNS responses. To also mitigate against active attackers pretending to be the server, the client must authenticate the (D)TLS connection to the server.

This document discusses Usage Profiles, which provide differing levels of attack mitigation to DNS clients, based on the requirements for authentication and encryption, regardless of the context (for example, which network the client is connected to). A Usage Profile is a distinct concept to a usage policy or usage model, which might dictate which Profile should be used in a particular context (enterprise vs coffee shop), with a particular set of DNS Servers or with reference to other external factors. A description of the variety of usage policies is out of scope of this document, but may be the subject of future work.

The term 'privacy-enabling DNS server' is used throughout this document. This is a DNS server that:

- o MUST implement DNS-over-TLS [RFC7858].
- o MAY implement DNS-over-DTLS [RFC8094].
- o SHOULD offer at least one of the credentials described in Section 8.
- o Implements the (D)TLS profile described in Section 9.

5. Usage Profiles

A DNS client has a choice of Usage Profiles available to increase the privacy of DNS transactions. This choice is briefly discussed in both [RFC7858] and [RFC8094]. These Usage Profiles are:

- o Strict profile: the DNS client requires both an encrypted and authenticated connection to a privacy-enabling DNS Server. A hard failure occurs if this is not available. This requires the client to securely obtain authentication information it can use to authenticate the server. This profile mitigates against both passive and active attacks providing the client with the best

available privacy for DNS. This Profile is discussed in detail in Section 6.6.

- o Opportunistic Privacy: the DNS client uses Opportunistic Security as described in [RFC7435]

"... the use of cleartext as the baseline communication security policy, with encryption and authentication negotiated and applied to the communication when available."

As described in [RFC7435] it might result in

- * an encrypted and authenticated connection
- * an encrypted connection
- * a clear text connection

depending on the fallback logic of the client, the available authentication information and the capabilities of the DNS Server. In all these cases the DNS client is willing to continue with a connection to the DNS Server and perform resolution of queries. The use of Opportunistic Privacy is intended to support incremental deployment of increased privacy with a view to widespread adoption of the Strict profile. It should be employed when the DNS client might otherwise settle for cleartext; it provides the maximum protection available depending on the combination of factors described above. If all the configured DNS Servers are DNS Privacy servers then it provides protection against passive attacks but not active ones.

Both profiles can include an initial meta query (performed using an Opportunistic lookup) to obtain the IP address for the privacy-enabling DNS server to which the DNS client will subsequently connect. The rationale for permitting this for the Strict profile is that requiring such meta queries to also be performed using the Strict profile would introduce significant deployment obstacles. However, it should be noted that in this scenario an active attack is possible on the meta query. Such an attack could result in a Strict profile client connecting to a server it cannot authenticate and so not obtaining DNS service, or an Opportunistic Privacy client connecting to a server controlled by the attacker. DNSSEC validation can detect the attack on the meta query and results in the client not obtaining DNS service (for both Usage profiles) because it will not proceed to connect to the server in question (see Section 7.2)

To compare the two Usage profiles the table below shows a successful Strict profile along side the 3 possible outcomes of an Opportunistic

profile. In the best case scenario for the Opportunistic profile (an authenticated and encrypted connection) it is equivalent to the Strict profile. In the worst case scenario it is equivalent to clear text. Clients using the Opportunistic profile SHOULD try for the best case but MAY fallback to the intermediate case and eventually the worst case scenario in order to obtain a response. One reason to fallback without trying every available privacy-enabling DNS server is if latency is more important than attack mitigation, see Appendix A. The Opportunistic profile therefore provides varying protection depending on what kind of connection is actually used including no attack mitigation at all.

Note that there is no requirement in Opportunistic Security to notify the user what type of connection is actually used, the 'detection' described below is only possible if such connection information is available. However, if it is available and the user is informed that an unencrypted connection was used to connect to a server then the user should assume (detect) that the connection is subject to both active and passive attack since the DNS queries are sent in clear text. This might be particularly useful if a new connection to a certain server is unencrypted when all previous connections were encrypted. Similarly if the user is informed that an encrypted but unauthenticated connection was used then the user can detect that the connection may be subject to active attack. In other words for the cases where no protection is provided against an attacker (N) it is possible to detect that an attack might be happening (D). This is discussed in Section 6.5.

Usage Profile	Connection	Passive Attacker	Active Attacker
Strict	A, E	P	P
Opportunistic	A, E	P	P
Opportunistic	E	P	N, D
Opportunistic		N, D	N, D

P == Protection; N == No protection; D == Detection is possible; A == Authenticated connection; E == Encrypted connection

Table 1: Attack protection by Usage Profile and type of attacker

The Strict profile provides the best attack mitigation and therefore SHOULD always be implemented in DNS clients that implement Opportunistic Privacy.

A DNS client that implements DNS-over-(D)TLS SHOULD NOT be configured by default to use only clear text.

The choice between the two profiles depends on a number of factors including which is more important to the particular client:

- o DNS service at the cost of no attack mitigation (Opportunistic) or
- o best available attack mitigation at the potential cost of no DNS service (Strict).

Additionally the two profiles require varying levels of configuration (or a trusted relationship with a provider) and DNS server capabilities, therefore DNS clients will need to carefully select which profile to use based on their communication needs.

A DNS server that implements DNS-over-(D)TLS SHOULD provide at least one credential so that those DNS clients that wish to do so are able to use the Strict profile (see Section 2).

5.1. DNS Resolution

A DNS client SHOULD select a particular Usage Profile when resolving a query. A DNS client MUST NOT fallback from Strict Privacy to Opportunistic Privacy during the resolution of a given query as this could invalidate the protection offered against attackers. It is anticipated that DNS clients will use a particular Usage Profile for all queries to all configured servers until an operational issue or policy update dictates a change in the profile used.

6. Authentication in DNS-over (D)TLS

This section describes authentication mechanisms and how they can be used in either Strict or Opportunistic Privacy for DNS-over-(D)TLS.

6.1. DNS-over-(D)TLS Startup Configuration Problems

Many (D)TLS clients use PKIX authentication [RFC6125] based on an authentication domain name for the server they are contacting. These clients typically first look up the server's network address in the DNS before making this connection. Such a DNS client therefore has a bootstrap problem, as it will typically only know the IP address of its DNS server.

In this case, before connecting to a DNS server, a DNS client needs to learn the authentication domain name it should associate with the IP address of a DNS server for authentication purposes. Sources of authentication domain names are discussed in Section 7.

One advantage of this domain name based approach is that it encourages association of stable, human recognizable identifiers with secure DNS service providers.

6.2. Credential Verification

The use of SPKI pinset verification is discussed in [RFC7858].

In terms of domain name based verification, once an authentication domain name is known for a DNS server a choice of authentication mechanisms can be used for credential verification. Section 8 discusses these mechanisms in detail, namely PKIX certificate based authentication and DANE.

Note that the use of DANE adds requirements on the ability of the client to get validated DNSSEC results. This is discussed in more detail in Section 8.2.

6.3. Summary of Authentication Mechanisms

This section provides an overview of the various authentication mechanisms. The table below indicates how the DNS client obtains information to use for authentication for each option; either statically via direct configuration or dynamically. Of course, the Opportunistic Usage Profile does not require authentication and so a client using that profile may choose to connect to a privacy-enabling DNS server on the basis of just an IP address.

#	Static Config	Dynamically Obtained	Short name: Description
1	SPKI + IP		SPKI: SPKI pinset(s) and IP address obtained out of band [RFC7858]
2	ADN + IP		ADN: ADN and IP address obtained out of band (see Section 7.1)
3	ADN	IP	ADN only: Opportunistic lookups to a NP DNS server for A/AAAA (see Section 7.2)
4		ADN + IP	DHCP: DHCP configuration only (see Section 7.3.1)
5	[ADN + IP]	[ADN + IP] TLSA record	DANE: DNSSEC chain obtained via Opportunistic lookups to NP DNS server (see Section 8.2.1)
6	[ADN + IP]	[ADN + IP] TLSA record	TLS extension: DNSSEC chain provided by PE DNS server in TLS DNSSEC chain extension (see Section 8.2.2)

SPKI == SPKI pinset(s), IP == IP Address, ADN == Authentication Domain Name, NP == Network provided, PE == Privacy-enabling, [] == Data may be obtained either statically or dynamically

Table 2: Overview of Authentication Mechanisms

The following summary attempts to present some key attributes of each of the mechanisms (using the 'Short name' from Table 2), indicating attractive attributes with a '+' and undesirable attributes with a '-'.

1. SPKI

+ Minimal leakage (Note that the ADN is always leaked in the Server Name Indication (SNI) field in the Client Hello in TLS when communicating with a privacy-enabling DNS server)

- Overhead of on-going key management required

2. ADN

- + Minimal leakage
- + One-off direct configuration only

3. ADN only

- + Minimal one-off direct configuration, only a human recognizable domain name needed
- A/AAAA meta queries leaked to network provided DNS server that may be subject to active attack (attack can be mitigated by DNSSEC validation).

4. DHCP

- + No static config
- Requires a non-standard or future DHCP option in order to provide the ADN
- Requires secure and trustworthy connection to DHCP server if used with a Strict Usage profile

5. DANE

The ADN and/or IP may be obtained statically or dynamically and the relevant attributes of that method apply

- + DANE options (e.g., matching on entire certificate)
- Requires a DNSSEC validating stub implementation (deployment of which is limited at the time of writing)
- DNSSEC chain meta queries leaked to network provided DNS server that may be subject to active attack

6. TLS extension

The ADN and/or IP may be obtained statically or dynamically and the relevant attributes of that method apply

- + Reduced latency compared with 'DANE'
- + No network provided DNS server required if ADN and IP statically configured
- + DANE options (e.g., matching on entire certificate)

- Requires a DNSSEC validating stub implementation

6.4. Combining Authentication Mechanisms

This draft does not make explicit recommendations about how an SPKI pinset based authentication mechanism should be combined with a domain based mechanism from an operator perspective. However it can be envisaged that a DNS server operator may wish to make both an SPKI pinset and an authentication domain name available to allow clients to choose which mechanism to use. Therefore, the following is guidance on how clients ought to behave if they choose to configure both, as is possible in HPKP [RFC7469].

A DNS client that is configured with both an authentication domain name and a SPKI pinset for a DNS server SHOULD match on both a valid credential for the authentication domain name and a valid SPKI pinset (if both are available) when connecting to that DNS server. In this case the client SHOULD treat the SPKI pin as specified in Section 2.6 of [RFC7469] with regard to user defined trust anchors. The overall authentication result SHOULD only be considered successful if both authentication mechanisms are successful.

6.5. Authentication in Opportunistic Privacy

An Opportunistic Security [RFC7435] profile is described in [RFC7858] which MAY be used for DNS-over-(D)TLS.

DNS clients issuing queries under an opportunistic profile and which know authentication information for a given privacy-enabling DNS server SHOULD try to authenticate the server using the mechanisms described here. This is useful for detecting (but not preventing) active attack, since the fact that authentication information is available indicates that the server in question is a privacy-enabling DNS server to which it should be possible to establish an authenticated and encrypted connection. In this case, whilst a client cannot know the reason for an authentication failure, from a security standpoint the client should consider an active attack in progress and proceed under that assumption. For example, a client that implements a nameserver selection algorithm that preferentially uses nameservers which successfully authenticated (see Section 5) might not continue to use the failing server if there were alternative servers available.

Attempting authentication is also useful for debugging or diagnostic purposes if there are means to report the result. This information can provide a basis for a DNS client to switch to (preferred) Strict Privacy where it is viable e.g, where all the configured servers support DNS-over-(D)TLS and successfully authenticate.

6.6. Authentication in Strict Privacy

To authenticate a privacy-enabling DNS server, a DNS client needs to know authentication information for each server it is willing to contact. This is necessary to protect against active attacks which attempt to re-direct clients to rogue DNS servers.

A DNS client requiring Strict Privacy MUST either use one of the sources listed in Section 7 to obtain an authentication domain name for the server it contacts, or use an SPKI pinset as described in [RFC7858].

A DNS client requiring Strict Privacy MUST only attempt to connect to DNS servers for which at least one piece of authentication information is known. The client MUST use the available verification mechanisms described in Section 8 to authenticate the server, and MUST abort connections to a server when no verification mechanism succeeds.

With Strict Privacy, the DNS client MUST NOT commence sending DNS queries until at least one of the privacy-enabling DNS servers becomes available.

A privacy-enabling DNS server may be temporarily unavailable when configuring a network. For example, for clients on networks that require registration through web-based login (a.k.a. "captive portals"), such registration may rely on DNS interception and spoofing. Techniques such as those used by DNSSEC-trigger [dnssec-trigger] MAY be used during network configuration, with the intent to transition to the designated privacy-enabling DNS servers after captive portal registration. If using a Strict Usage profile the system MUST alert by some means that the DNS is not private during such bootstrap.

6.7. Implementation guidance

Section 9 describes the (D)TLS profile for DNS-over(D)TLS. Additional considerations relating to general implementation guidelines are discussed in both Section 11 and in Appendix A.

7. Sources of Authentication Domain Names

7.1. Full direct configuration

DNS clients may be directly and securely provisioned with the authentication domain name of each privacy-enabling DNS server. For example, using a client specific configuration file or API.

In this case, direct configuration for a DNS client would consist of both an IP address and an authentication domain name for each DNS server.

7.2. Direct configuration of ADN only

A DNS client may be configured directly and securely with only the authentication domain name of each of its privacy-enabling DNS servers. For example, using a client specific configuration file or API.

A DNS client might learn of a default recursive DNS resolver from an untrusted source (such as DHCP's DNS server option [RFC3646]). It can then use Opportunistic DNS connections to an untrusted recursive DNS resolver to establish the IP address of the intended privacy-enabling DNS resolver by doing a lookup of A/AAAA records. Such records SHOULD be DNSSEC validated when using a Strict Usage profile and MUST be validated when using Opportunistic Privacy. Private DNS resolution can now be done by the DNS client against the pre-configured privacy-enabling DNS resolver, using the IP address gathered from the untrusted DNS resolver.

A DNS client so configured that successfully connects to a privacy-enabling DNS server MAY choose to locally cache the server host IP addresses in order to not have to repeat the opportunistic lookup.

7.3. Dynamic discovery of ADN

This section discusses the general case of a DNS client discovering both the authentication domain name and IP address dynamically. This is not possible at the time of writing by any standard means. However since, for example, a future DHCP extension could (in principle) provide this mechanism the required security properties of such mechanisms are outlined here.

When using a Strict profile the dynamic discovery technique used as a source of authentication domain names MUST be considered secure and trustworthy. This requirement does not apply when using an Opportunistic profile given the security expectation of that profile.

7.3.1. DHCP

In the typical case today, a DHCP server [RFC2131] [RFC3315] provides a list of IP addresses for DNS resolvers (see Section 3.8 of [RFC2132]), but does not provide an authentication domain name for the DNS resolver, thus preventing the use of most of the authentication methods described here (all those that are based on a mechanism with ADN in Table 2).

This document does not specify or request any DHCP extension to provide authentication domain names. However, if one is developed in future work the issues outlined in Section 8 of [RFC7227] should be taken into account as should the Security Considerations in Section 23 of [RFC3315]).

This document does not attempt to describe secured and trusted relationships to DHCP servers, which is a purely DHCP issue (still open, at the time of writing.) Whilst some implementation work is in progress to secure IPv6 connections for DHCP, IPv4 connections have received little to no implementation attention in this area.

8. Authentication Domain Name based Credential Verification

8.1. PKIX Certificate Based Authentication

When a DNS client configured with an authentication domain name connects to its configured DNS server over (D)TLS, the server may present it with a PKIX certificate. In order to ensure proper authentication, DNS clients MUST verify the entire certification path per [RFC5280]. The DNS client additionally uses [RFC6125] validation techniques to compare the domain name to the certificate provided.

A DNS client constructs one Reference Identifier for the server based on the authentication domain name: A DNS-ID which is simply the authentication domain name itself.

If the Reference Identifier is found in the PKIX certificate's subjectAltName extension as described in section 6 of [RFC6125], the DNS client should accept the certificate for the server.

A compliant DNS client MUST only inspect the certificate's subjectAltName extension for the Reference Identifier. In particular, it MUST NOT inspect the Subject field itself.

8.2. DANE

DANE [RFC6698] provides mechanisms to anchor certificate and raw public key trust with DNSSEC. However this requires the DNS client to have an authentication domain name for the DNS Privacy Server which must be obtained via a trusted source.

This section assumes a solid understanding of both DANE [RFC6698] and DANE Operations [RFC7671]. A few pertinent issues covered in these documents are outlined here as useful pointers, but familiarity with both these documents in their entirety is expected.

It is noted that [RFC6698] says

"Clients that validate the DNSSEC signatures themselves MUST use standard DNSSEC validation procedures. Clients that rely on another entity to perform the DNSSEC signature validation MUST use a secure mechanism between themselves and the validator."

It is noted that [RFC7671] covers the following topics:

- o Section 4.1: Opportunistic Security and PKIX Usages and Section 14: Security Considerations, which both discuss the use of Trust Anchor and End Entity based schemes (PKIX-TA(0) and PKIX-EE(1) respectively) for Opportunistic Security.
- o Section 5: Certificate-Usage-Specific DANE Updates and Guidelines. Specifically Section 5.1 which outlines the combination of Certificate Usage DANE-EE(3) and Selector Usage SPKI(1) with Raw Public Keys [RFC7250]. Section 5.1 also discusses the security implications of this mode, for example, it discusses key lifetimes and specifies that validity period enforcement is based solely on the TLSA RRset properties for this case.
- o Section 13: Operational Considerations, which discusses TLSA TTLS and signature validity periods.

The specific DANE record for a DNS Privacy Server would take the form:

_853._tcp.[authentication-domain-name] for TLS

_853._udp.[authentication-domain-name] for DTLS

8.2.1. Direct DNS Lookup

The DNS client MAY choose to perform the DNS lookups to retrieve the required DANE records itself. The DNS queries for such DANE records MAY use Opportunistic encryption or be in the clear to avoid trust recursion. The records MUST be validated using DNSSEC as described above in [RFC6698].

8.2.2. TLS DNSSEC Chain extension

The DNS client MAY offer the TLS extension described in [I-D.ietf-tls-dnssec-chain-extension]. If the DNS server supports this extension, it can provide the full chain to the client in the handshake.

If the DNS client offers the TLS DNSSEC Chain extension, it MUST be capable of validating the full DNSSEC authentication chain down to the leaf. If the supplied DNSSEC chain does not validate, the client

MUST ignore the DNSSEC chain and validate only via other supplied credentials.

9. (D)TLS Protocol Profile

This section defines the (D)TLS protocol profile of DNS-over-(D)TLS.

Clients and servers MUST adhere to the (D)TLS implementation recommendations and security considerations of [RFC7525] except with respect to (D)TLS version.

Since encryption of DNS using (D)TLS is a green-field deployment DNS clients and servers MUST implement only (D)TLS 1.2 or later. For example, implementing TLS 1.3 [I-D.ietf-tls-tls13] is also an option.

Implementations MUST NOT offer or provide TLS compression, since compression can leak significant amounts of information, especially to a network observer capable of forcing the user to do an arbitrary DNS lookup in the style of the CRIME attacks [CRIME].

Implementations compliant with this profile MUST implement all of the following items:

- o TLS session resumption without server-side state [RFC5077] which eliminates the need for the server to retain cryptographic state for longer than necessary (This statement updates [RFC7858]).
- o Raw public keys [RFC7250] which reduce the size of the ServerHello, and can be used by servers that cannot obtain certificates (e.g., DNS servers on private networks). A client MUST only indicate support for raw public keys if it has an SPKI pinset pre-configured (for interoperability reasons).

Implementations compliant with this profile SHOULD implement all of the following items:

- o TLS False Start [RFC7918] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the (encrypted) DNS query.
- o Cached Information Extension [RFC7924] which avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake.

Guidance specific to TLS is provided in [RFC7858] and that specific to DTLS it is provided in [RFC8094].

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

Security considerations discussed in [RFC7525], [RFC8094] and [RFC7858] apply to this document.

DNS Clients SHOULD implement support for the mechanisms described in Section 8.2 and offering a configuration option which limits authentication to using only those mechanisms (i.e., with no fallback to pure PKIX based authentication) such that authenticating solely via the PKIX infrastructure can be avoided.

11.1. Counter-measures to DNS Traffic Analysis

This section makes suggestions for measures that can reduce the ability of attackers to infer information pertaining to encrypted client queries by other means (e.g., via an analysis of encrypted traffic size, or via monitoring of resolver to authoritative traffic).

DNS-over-(D)TLS clients and servers SHOULD implement the following relevant DNS extensions

- o EDNS(0) padding [RFC7830], which allows encrypted queries and responses to hide their size making analysis of encrypted traffic harder.

Guidance on padding policies for EDNS(0) is provided in [I-D.ietf-dprive-padding-policy]

DNS-over-(D)TLS clients SHOULD implement the following relevant DNS extensions

- o Privacy Election using Client Subnet in DNS Queries [RFC7871]. If a DNS client does not include an EDNS0 Client Subnet Option with a SOURCE PREFIX-LENGTH set to 0 in a query, the DNS server may potentially leak client address information to the upstream authoritative DNS servers. A DNS client ought to be able to inform the DNS Resolver that it does not want any address information leaked, and the DNS Resolver should honor that request.

12. Acknowledgments

Thanks to the authors of both [RFC8094] and [RFC7858] for laying the ground work that this draft builds on and for reviewing the contents. The authors would also like to thank John Dickinson, Shumon Huque, Melinda Shore, Gowri Visweswaran, Ray Bellis, Stephane Bortzmeyer, Jinmei Tatuya, Paul Hoffman, Christian Huitema and John Levine for review and discussion of the ideas presented here.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <<https://www.rfc-editor.org/info/rfc5077>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.

- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <<https://www.rfc-editor.org/info/rfc7671>>.
- [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <<https://www.rfc-editor.org/info/rfc7830>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <<https://www.rfc-editor.org/info/rfc7918>>.
- [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <<https://www.rfc-editor.org/info/rfc7924>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.

13.2. Informative References

- [CRIME] Rizzo, J. and T. Duong, "The CRIME Attack", 2012.
- [dnssec-trigger]
NLnetLabs, "Dnssec-Trigger", May 2014,
<<https://www.nlnetlabs.nl/projects/dnssec-trigger/>>.
- [I-D.ietf-dprive-padding-policy]
Mayrhofer, A., "Padding Policy for EDNS(0)", draft-ietf-dprive-padding-policy-01 (work in progress), July 2017.
- [I-D.ietf-tls-dnssec-chain-extension]
Shore, M., Barnes, R., Huque, S., and W. Toorop, "A DANE Record and DNSSEC Authentication Chain Extension for TLS", draft-ietf-tls-dnssec-chain-extension-04 (work in progress), June 2017.
- [I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-21 (work in progress), July 2017.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997,
<<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,
<<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014,
<<https://www.rfc-editor.org/info/rfc7227>>.

- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<https://www.rfc-editor.org/info/rfc7469>>.
- [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <<https://www.rfc-editor.org/info/rfc7626>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.

Appendix A. Server capability probing and caching by DNS clients

This section presents a non-normative discussion of how DNS clients might probe for and cache capabilities of privacy-enabling DNS servers.

Deployment of both DNS-over-TLS and DNS-over-DTLS will be gradual. Not all servers will support one or both of these protocols and the well-known port might be blocked by some middleboxes. Clients will be expected to keep track of servers that support DNS-over-TLS and/or DNS-over-DTLS, and those that have been previously authenticated.

If no server capability information is available then (unless otherwise specified by the configuration of the DNS client) DNS clients that implement both TLS and DTLS should try to authenticate using both protocols before failing or falling back to a unauthenticated or clear text connections. DNS clients using an Opportunistic Usage profile should try all available servers (possibly in parallel) in order to obtain an authenticated and encrypted connection before falling back. (RATIONALE: This approach can increase latency while discovering server capabilities but maximizes the chance of sending the query over an authenticated and encrypted connection.)

Appendix B. Changes between revisions

[Note to RFC Editor: please remove this section prior to publication.]

B.1. -11 version

Section 5: Re-ordered and re-worded the text in section on Opportunistic profile to make the protection offered by Opportunistic clearer.

Section 5: Provide a more detailed analysis of attacks on the meta queries

Section 7.2: Re-introduce a requirement to DNSSEC validate the meta-queries making it as SHOULD for Strict and a MUST for Opportunistic.

B.2. -10 version

Clarified the specific attacks the Usage profiles mitigate against.

Revised wording in the draft relating 'security/privacy guarantees' and generally improved consistency of wording throughout the document.

Corrected and added a number of references:

- o RFC7924 is now Normative
- o RFC7918 and RFC8094 are now Normative (and therefore Downrefs)
- o draft-ietf-tls-tls13, draft-ietf-dprive-padding-policy, RFC3315 and RFC7227 added

Terminology: Update definition of Privacy-enabling DNS server and moved normative definition to section 4.

Section 5 and 6.3: Included discussion of the additional attacks possible when using meta-queries to bootstrap the DNS service

Section 5: Added sentence on why Opportunistic Profile may fallback for latency reasons.

Section 5.1: Added discussion of when clients might change Usage Profiles.

Section 6.4: Added caveat on use of combined authentication re RFC7469.

Section 6.5: Added more detail on how authentication results might be used in Opportunistic. Opportunistic clients now SHOULD try for the best case.

Section 7.3: Re-worked this section and the discussion of DHCP.

Section 9: Removed unnecessary text, added condition on use of RFC7250 (Raw public keys).

Section 11.: More detail on padding policies.

Numerous editorial corrections.

B.3. -09 version

Remove the SRV record to simplify the draft.

Add suggestion that clients offer option to avoid using only PKIX authentication.

Clarify that the MUST on implementing TLS session resumption updates RFC7858.

Update page header to be '(D)TLS Authentication for TLS'.

B.4. -08 version

Removed hard failure as an option for Opportunistic Usage profile.

Added a new section comparing the Authentication Mechanisms

B.5. -07 version

Re-work of the Abstract and Introduction to better describe the contents in this version.

Terminology: New definition of 'authentication information'.

Scope: Changes to the Scope section.

Moved discussion of combining authentication mechanism earlier.

Changes to the section headings and groupings to make the presentation more logical.

B.6. -06 version

Introduction: Re-word discussion of Working group charter.

Introduction: Re-word first and third bullet point about 'obtaining' a domain name and IP address.

Introduction: Update reference to DNS-over-TLS draft.

Terminology: Change forwarder/proxy to just forwarder

Terminology: Add definition of 'Authentication domain name' and use this throughout

Section 4.2: Remove parenthesis in the table.

Section 4.2: Change the text after the table as agreed with Paul Hoffman.

Section 4.3.1: Change title and remove brackets around last statement.

Section 11: Split second paragraph.

B.7. -05 version

Add more details on detecting passive attacks to section 4.2

Changed X.509 to PKIX throughout

Change comment about future I-D on usage policies.

B.8. -04 version

Introduction: Add comment that DNS-over-DTLS draft is Experiments

Update 2 I-D references to RFCs.

B.9. -03 version

Section 9: Update DANE section with better references to RFC7671 and RFC7250

B.10. -02 version

Introduction: Added paragraph on the background and scope of the document.

Introduction and Discussion: Added more information on what a Usage profiles is (and is not) the the two presented here.

Introduction: Added paragraph to make a comparison with the Strict profile in RFC7858 clearer.

Section 4.2: Re-worked the description of Opportunistic and the table.

Section 8.3: Clarified statement about use of DHCP in Opportunistic profile

Title abbreviated.

B.11. -01 version

Section 4.2: Make clear that the Strict Privacy Profile can include meta queries performed using Opportunistic Privacy.

Section 4.2, Table 1: Update to clarify that Opportunistic Privacy does not guarantee protection against passive attack.

Section 4.2: Add sentence discussing client/provider trusted relationships.

Section 5: Add more discussion of detection of active attacks when using Opportunistic Privacy.

Section 8.2: Clarify description and example.

B.12. draft-ietf-dprive-dtls-and-tls-profiles-00

Re-submission of draft-dgr-dprive-dtls-and-tls-profiles with name change to draft-ietf-dprive-dtls-and-tls-profiles. Also minor nits fixed.

Authors' Addresses

Sara Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
UK

Email: sara@sinodun.com
URI: <http://sinodun.com>

Daniel Kahn Gillmor
ACLU
125 Broad Street, 18th Floor
New York NY 10004
USA

Email: dkg@fifthhorseman.net

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 20, 2019

A. Mayrhofer
nic.at GmbH
July 19, 2018

Padding Policy for EDNS(0)
draft-ietf-dprive-padding-policy-06

Abstract

RFC 7830 specifies the EDNS(0) 'Padding' option, but does not specify the actual padding length for specific applications. This memo lists the possible options ("Padding Policies"), discusses implications of each of these options, and provides a recommended (experimental) option.

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 January 20, 2019.

Copyright Notice

Copyright (c) 2018 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. General Guidance	3
4. Padding Strategies	3
4.1. Block Length Padding - Recommended Strategy	3
4.2. Other Strategies	5
4.2.1. Maximal Length Padding	5
4.2.2. Random Length Padding	5
4.2.3. Random Block Length Padding	6
5. Acknowledgements	6
6. IANA Considerations	6
7. Security Considerations	7
8. Changes	7
8.1. draft-ietf-dprive-padding-policy-06	7
8.2. draft-ietf-dprive-padding-policy-05	7
8.3. draft-ietf-dprive-padding-policy-04	8
8.4. draft-ietf-dprive-padding-policy-03	8
8.5. draft-ietf-dprive-padding-policy-02	8
8.6. draft-ietf-dprive-padding-policy-01	8
8.7. draft-ietf-dprive-padding-policy-00	8
8.8. draft-mayrhofer-dprive-padding-profiles-00	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Appendix A. Non-sensible Padding Policies	9
A.1. No Padding	9
A.2. Fixed Length Padding	10
Author's Address	10

1. Introduction

[RFC7830] specifies the Extensions Mechanisms for DNS (EDNS(0)) "Padding" option, which allows DNS clients and servers to artificially increase the size of a DNS message by a variable number of bytes, hampering size-based correlation of encrypted DNS messages.

However, RFC 7830 deliberately does not specify the actual length of padding to be used. This memo discusses options regarding the actual size of padding, lists advantages and disadvantages of each of these "Padding Strategies", and provides a recommended (experimental) strategy.

Padding DNS messages is useful only when transport is encrypted, using protocols such as DNS over Transport Layer Security [RFC7858], DNS over Datagram Transport Layer Security [RFC8094] or other encrypted DNS transports specified in the future.

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.

3. General Guidance

EDNS(0) options space: The maximum message length as dictated by the protocol limits the space for EDNS(0) options. Since padding will reduce the message space available to other EDNS(0) options, "Padding" MUST be the last EDNS(0) option applied before a DNS message is sent.

Resource Conservation: Especially in situations where networking and processing resources are scarce (e.g. battery powered long-life devices, low bandwidth or high cost links), the tradeoff between increased size of padded DNS messages and the corresponding gain in confidentiality must be carefully considered.

Transport Protocol Independence: The message size used as input to the various padding strategies MUST be calculated excluding the potential extra 2-octet length field used in TCP transport. Otherwise, the padded (observable) size of the DNS packets could significantly change between different transport protocols, and reveal an indication of the original (unpadded) length. For example, given a "Block Length" padding strategy with a block length of 32 octets, and a DNS message with a size of 59 octets, the message would be padded to 64 octets when transported over UDP. If that same message was transported over TCP, and the padding strategy would consider the extra 2 octets of the length field (61 octets in total), the padded message would be 96 octets long (as the minimum length of the Padding option is 4 octets).

4. Padding Strategies

This section contains a recommended strategy, as well as a non-exhaustive list of other sensible strategies in choosing padding length. Note that, for completeness, Appendix A contains two more (non-sensible) strategies.

4.1. Block Length Padding - Recommended Strategy

Based on empirical research performed by Daniel K. Gillmor [dkg-padding-ndss], EDNS Padding SHOULD be performed following the "Block Length Padding" strategy as follows:

- (1) Clients SHOULD pad queries to the closest multiple of 128 octets.
- (2) If a Server receives a query that includes the EDNS(0) Padding Option, it MUST pad the corresponding response (See Section 4 of RFC7830) and SHOULD pad the corresponding response to a multiple of 468 octets (see below).

Note that the recommendation above applies only if the DNS transport is encrypted (See Section 6 of RFC 7830).

In Block Length Padding, a sender pads each message so that its padded length is a multiple of a chosen block length. This creates a greatly reduced variety of message lengths. An implementor needs to consider that even the zero-length EDNS(0) Padding Option increases the length of the packet by 4 octets.

Options: Block Length - for queries, values between 16 and 128 octets were discussed before empiric research was performed. Responses will require larger block sizes (see [dkg-padding-ndss] and above for a discussion).

Very large block lengths will have confidentiality properties similar to the "Maximal Length Padding" strategy (Section 4.2.1), since almost all messages will fit into a single block. Such "very large block length" values are 288 bytes for the query (the maximum size of a one-question query over TCP, without any EDNS(0) options), and the EDNS(0) buffer size of the server for the responses.

Advantages: This policy is reasonably easy to implement, reduces the variety of message ("fingerprint") sizes significantly, and does not require a source of (pseudo) random numbers, since the padding length required can be derived from the actual (unpadded) message.

Disadvantage: Given an unpadded message and the block size of the padding (which is assumed to be public knowledge once a server is reachable), the size range of a padded message can be predicted. Therefore, the minimum length of the unpadded message can be inferred.

The empirical research cited above performed a simulation of padding, based on real-world DNS traffic captured on busy recursive resolvers of a research network. The evaluation of the performance of individual padding policies was based on a "cost to attacker" and "cost to defender" function, where the "cost to attacker" was defined as the percentage of query/response pairs falling into the same size bucket, and "cost to defender" as the size factor between padded and unpadded messages. Padding with a block size of 128 bytes on the query side, and 468 bytes on the response side was considered the

optimum trade-off between defender and attacker cost. The response block size of 468 was chosen so that 3 blocks of 468 octets would still comfortably fit into typical Maximum Transmission Unit (MTU) size values.

The Block Size will interact with the MTU size. Especially for length values that are a large fraction of the MTU, unless the block length is chosen so that a multiple just fits into the MTU, Block Padding may cause unnecessary fragmentation for UDP based delivery. Also, choosing a block length larger than the MTU of course always forces to always fragment.

Note: Once DNSSEC validating clients become more prevalent, observed size patterns are expected to change significantly. In such case, the recommended strategy might need to be revisited.

4.2. Other Strategies

4.2.1. Maximal Length Padding

In Maximal Length Padding the sender pads every message to the maximum size as allowed by protocol negotiations.

Advantages: Maximal Length Padding, when combined with encrypted transport, provides the highest possible level of message size confidentiality.

Disadvantages: Maximal Length Padding is wasteful, and requires resources on the client, all intervening network and equipment, and the server. Depending on the negotiated size, this strategy will commonly exceed the MTU, and then result in a consistent number of fragments reducing delivery probability when datagram based transport (such as UDP) is used.

Due to resource consumption, Maximal Length Padding is NOT RECOMMENDED.

4.2.2. Random Length Padding

When using Random Length Padding, a sender pads each message with a random amount of padding. Due to the size of the EDNS(0) Padding Option itself, each message size is hence increased by at least 4 octets. The upper limit for padding is the maximum message size. However, a client or server may choose to impose a lower maximum padding length.

Options: Maximum and minimum padding length.

Advantages: Theoretically, this policy should create a natural "distribution" of message sizes.

Disadvantage: Random Length padding allows an attacker who can observe a large number of requests to infer the length of the original value by observing the distribution of total lengths.

According to the limited empirical data available, Random Length Padding exposes slightly more entropy to an attacker than Block Length Padding. Due to that, and the risk outlined above, Random Length Padding is NOT RECOMMENDED.

4.2.3. Random Block Length Padding

This policy combines Block Length Padding with a random component. Specifically, a sender randomly chooses between a few block length values and then applies Block Length Padding based on the chosen block length. The random selection of block length might even be reasonably based on a "weak" source of randomness, such as the transaction ID of the message.

Options: Number of and the values for the set of Block Lengths, source of "randomness"

Advantages: Compared to Block Length Padding, this creates more variety in the resulting message sizes for a certain individual original message length.

Disadvantage: Requires more implementation effort compared to simple Block Length Padding

Random Block Length Padding (as other combinations of padding strategies) requires further empirical study.

5. Acknowledgements

Daniel K. Gillmor performed empirical research out of which the "Recommended Strategy" was copied. Stephane Bortzmeyer and Hugo Connery provided text. Shane Kerr, Sara Dickinson, Paul Hoffman, Magnus Westerlund, Charlie Kaufman, Joe Clarke and Meral Shirazipour performed reviews or provided substantial comments.

6. IANA Considerations

This document has no considerations for IANA.

7. Security Considerations

The choice of the right padding policy (and the right parameters for the chosen policy) has a significant impact on the resilience of encrypted DNS against size-based correlation attacks. Therefore, any implementor of EDNS(0) Padding must carefully consider which policies to implement, the default policy chosen, which parameters to make configurable, and the default parameter values.

No matter how carefully a client selects their Padding policy, this effort can be jeopardized if the server chooses to apply an ineffective Padding policy to the corresponding response packets. Therefore, a client applying Padding may want to choose a DNS server which does apply at least an equally effective Padding policy on responses.

Note that even with encryption and padding, it might be trivial to identify that the observed traffic is DNS. Also, padding does not prevent information leak via other side channels (particularly timing information and number of query/response pairs). Counter-measures against such other side channels could include injecting artificial "cover traffic" into the stream of DNS messages, or delaying DNS responses by a certain amount of jitter. Such strategies are out of scope of this document. Additionally, there is neither enough theoretic analysis nor experimental data available to recommend any such countermeasures.

8. Changes

[Note to RFC Editors: This whole section is to be removed before publication]

8.1. draft-ietf-dprive-padding-policy-06

Changes based on IESG evaluation: Removed duplicate paragraph about MTU impact, switched Terminology boilerplate to RFC8174, changed text regarding Random Padding, changed text regarding very large block paddings, some minor edits.

8.2. draft-ietf-dprive-padding-policy-05

Changes based on outcomes of IETF-wide LC + various reviews: Meral Shirazipour (Gen-ART), Charlie Kaufmann (SECDIR), Joe Clarke (OPSDIR - changed document flow based on comments),

8.3. draft-ietf-dprive-padding-policy-04

Changes based on WGLC: Changed implementor consideration text in Security Con section (Sara), moved "No Padding" and "Fixed Length Padding" to appendix (Stephane, Paul), Changed TODO in Random Padding to info from empirical study (Stephen), Added note to pad only if transport encrypted (Stephen), added intro text referencing to DNSoTLS and DNSoDTLS (Stephane), added text about timing/jitter to security considerations.

8.4. draft-ietf-dprive-padding-policy-03

Editorial changes in various spots. Added text about excluding TCP length field, more security considerations, addressing Sara's other feedback to -02.

8.5. draft-ietf-dprive-padding-policy-02

Changed Document Status to Experimental, added "maximum length" padding policy, reworded "block length" policy, some editorial changes.

8.6. draft-ietf-dprive-padding-policy-01

Some (mostly editorial) changes to text. Added "Recommendation" section based on dkg's research.

8.7. draft-ietf-dprive-padding-policy-00

Initial (mostly unmodified) WG version. Changed "Profile" to "Policy" to avoid confusion with the (D)TLS profiles document.

8.8. draft-mayrhofer-dprive-padding-profiles-00

Initial version

9. References

9.1. Normative References

[dkg-padding-ndss]

Gillmor, D., "Empirical DNS Padding Policy", March 2017, <<https://dns.cmrq.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <<https://www.rfc-editor.org/info/rfc7830>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.

Appendix A. Non-sensible Padding Policies

A.1. No Padding

In the "No Padding" policy, the EDNS0 Padding option is not used, and the size of the final (actually, "non-padded") message obviously exactly matches the size of the unpadded message. Even though this "non-policy" seems redundant in this list, its properties must be considered for cases where just one of the parties (client or server) applies padding.

Also, this "policy" is required when the remaining message size of the unpadded message does not allow for the Padding option to be included (less than 4 octets left).

Advantages: This "policy" requires no additional resources on client, server and network side.

Disadvantages: The original size of the message remains unchanged, hence this approach provides no additional confidentiality.

"No Padding" MUST NOT be used unless message size disallows the use of Padding.

A.2. Fixed Length Padding

In fixed length padding, a sender chooses to pad each message with a padding of constant length.

Options: Actual length of padding

Advantages: Since the padding is constant in length, this policy is very easy to implement, and at least ensures that the message length diverges from the length of the original packet (even only by a fixed value)

Disadvantage: Obviously, the amount of padding easily discoverable from a single unencrypted message, or by observing message patterns. When a public DNS server applies this policy, the length of the padding hence must be assumed to be public knowledge. Therefore, this policy is (almost) as useless as the "No Padding" option described above.

"Fixed Length Padding" MUST NOT be used except for test applications.

Author's Address

Alexander Mayrhofer
nic.at GmbH
Karlsplatz 1/2/9
Vienna 1010
Austria

Email: alex.mayrhofer.ietf@gmail.com
URI: <http://edns0-padding.org/>