DNS Extensions Working Group
Internet-Draft
Intended status: Standards Track

G. Barwood

7 April 2010

DNS Transport
draft-barwood-dnsext-dns-transport-18

Abstract

This document describes a new transport protocol for DNS.
IP fragmentation is avoided, blind spoofing, amplification attacks
and other denial of service attacks are prevented. Latency for a
typical DNS query is a single round trip, after a setup handshake.
No per-client server state is required between transactions.
Packets may optionally be encrypted and authenticated.
The protocol may have other applications.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on October 7, 2010.

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

DNSSEC implies that DNS responses may be large, possibly larger than the de facto ~1500 byte internet MTU.

Large responses are a challenge for DNS transport. EDNS [RFC2671] was introduced in 1999 to allow larger responses to be sent over UDP, previously DNS/UDP was limited to a 512 bytes.

EDNS is problematic for several reasons:

(1) It allows amplification attacks against 3rd parties. DNS/UDP has always been susceptible to these attacks, but EDNS has increased the amplification factor by an order of magnitude.

(2) The IP protocol specifies a means by which large IP packets are split into fragments and then re-assembled. However fragmented UDP responses are undesirable for several reasons:

- Fragments may be spoofed. The DNS ID and port number are only present in the first fragment, and the IP ID may be easy for an attacker to predict.

- In practice fragmentation is not reliable, and large UDP packets may fail to be delivered.

- If a single fragment is lost, the entire response must be re-sent.

- Re-assembling fragments requires buffer resources, which opens up denial of service attacks [GONT].

Instead, it is possible to use TCP, but this is undesirable, as TCP imposes increased latency and significant server state that may be vulnerable to denial of service attack.

Nearly all current DNS traffic is carried by UDP with a maximum size of 512 bytes, and relying on TCP is a risk for the deployment of DNSSEC.

Therefore a new protocol is proposed, with mnenomic QRP, to stand for "Quick Response Protocol".

2. Definitions and Objectives

2.1 Definitions

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].

DNS Payload A DNS Message [RFC1035], not including the 16-bit ID field. For AXFR, the response messages are concatenated without ID fields, to form a single DNS payload.

Transaction A transaction is initiated by a client request packet and the server responds with one or more response packets. All the packets in a transaction have the same REQUESTID.

Transfer The requested transfer of a DNS Payload, using one or more transactions as described in sections 3.6 and 3.7.

2.2 Objectives

Fragmentation must not occur provided the actual path MTU is at least the MTU sent by the client or 600 bytes, whichever is larger.

Blind spoofing attacks must be prevented. Amplification attacks against third parties must be prevented.

No per-client server state must be needed between transactions.

Each Transfer (for moderate response sizes) is performed in a single round trip, after setup.

The protocol should be efficient : only lost IP packets should be

re-transmitted.

3. Protocol

3.1 Overview

Communication is over UDP [RFC768] in two stages. First a long-lived SERVERTOKEN is acquired by the client. Subsequent queries are protected against amplification attacks by the SERVERTOKEN.

Each UDP packet starts with a 16 bit OPCODE, followed by a 12 byte REQUESTID that identifies the transaction. These fields are not shown in the packet diagrams.

Fixed length field sizes are as shown in the packet diagrams. All numbers are unsigned integers, with the first bit being the most significant.

Variable length reserved areas MUST be omitted by the sender.

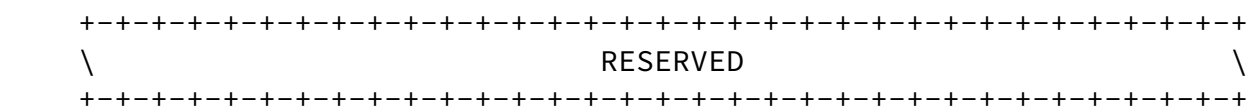
Fixed length reserved areas MUST be set to zero by the sender.

All reserved areas MUST be ignored by the receiver.

Parameters are stored in DS records, as described in section 3.12.

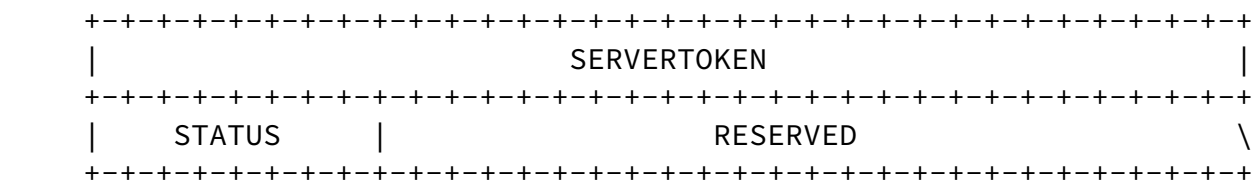
3.2 Setup request

The client acquires a SERVERTOKEN for a given Server IP address by sending a packet with OPCODE 1, format :



3.3 Setup response

The server response has OPCODE 1, format :



where :

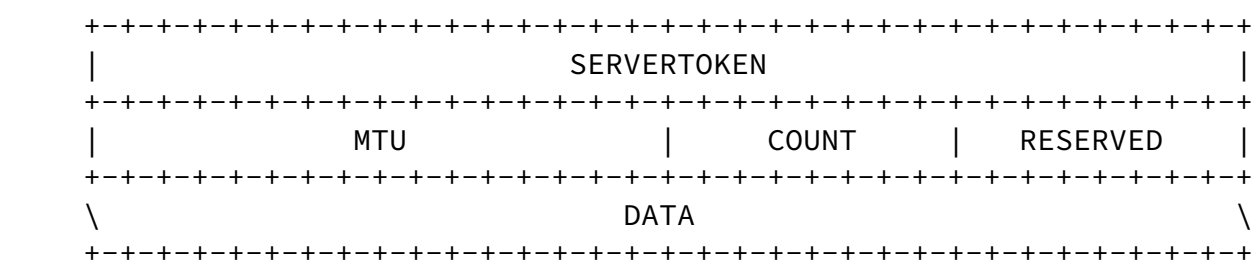
SERVERTOKEN is a 32 bit value computed as a secure hash of the client IP Address and a long term server secret. Servers MUST change the long term secret at least once every 4 weeks.

STATUS is an 8 bit status code, see section 3.10.

The client associates SERVERTOKEN, and the client IP address (for multi-homed clients) with the Server IP address.

3.4 Initial request

To make a DNS request, a packet is sent with OPCODE 2, format:



where :

SERVERTOKEN is a copy of SERVERTOKEN from the setup response.

MTU limits the size in bytes of the IP packets used to send the response. MUST be at least 600.

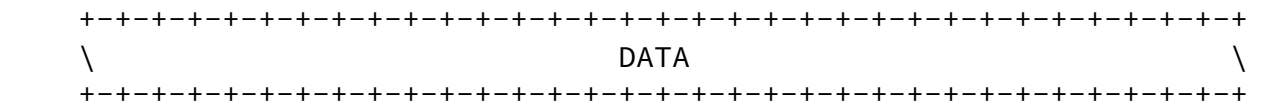
COUNT limits the number of pages the server will send.

DATA is the DNS payload.

3.5 Server response : single page

The server checks SERVERTOKEN, and obtains the DNS response payload. If the requested MTU is less than 600 bytes, the server SHOULD set MTU to 600 bytes. If the path MTU is known to be less than the value supplied by the client, MTU is reduced to that value (but not to less than 600 bytes).

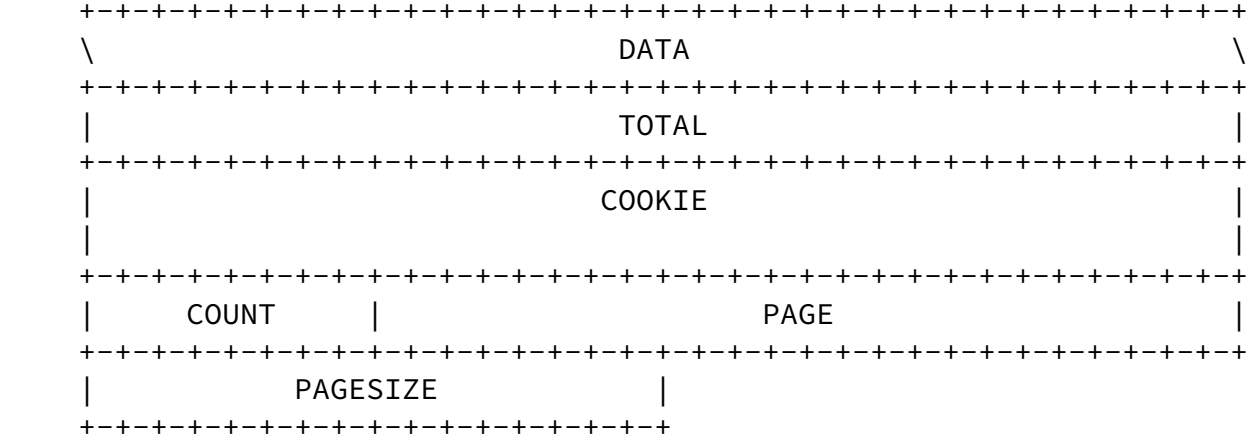
If the DNS payload size plus IP/UDP/QRP overhead is not greater than MTU, the server sends a single page response, OPCODE 2, format :



where DATA is the DNS payload. The client uses DATA as the normal DNS response.

3.6 Server response : multi page

Otherwise, the server divides the DNS payload into equal size pages (except for the last page which may be smaller), so that each IP response packet does not exceed MTU, and sends multiple packets, each with OPCODE 3 and format :



where :

DATA is part of the DNS payload.

TOTAL is the size of the complete DNS payload.

COOKIE is a 64-bit value used to request further pages.

COUNT is the number of pages sent.

PAGE is the 24-bit zero-based number of this page.

PAGESIZE is the size into which the DNS payload has been divided.

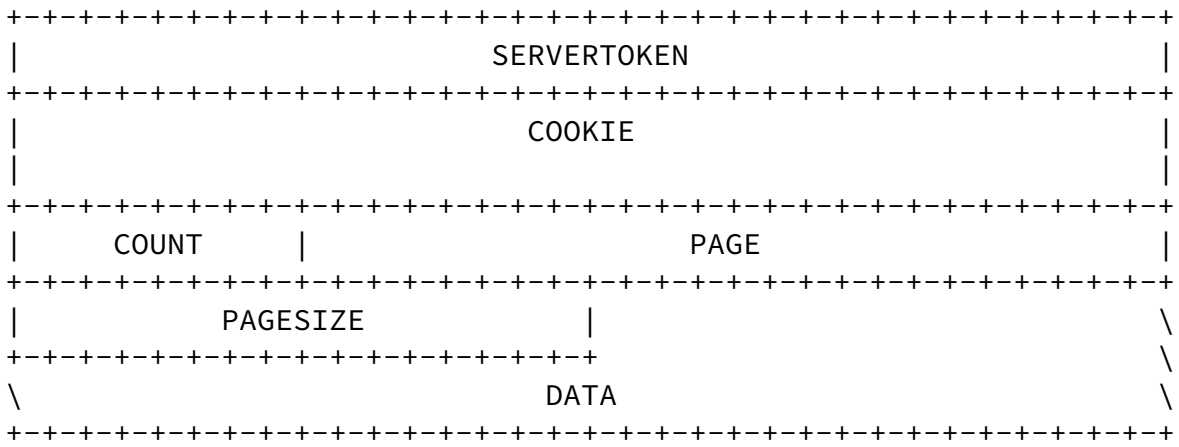
The client allocates an assembly buffer of TOTAL bytes (if not already allocated), and copies DATA into it at offset PAGE x PAGESIZE.

Clients SHOULD impose limits on the maximum size response (TOTAL) they will accept, to prevent attacks by malicious servers.

Servers MAY send a smaller number of pages than requested, for policy reasons, or if there is local congestion. The pages sent have numbers 0 .. COUNT-1.

3.7 Follow-up request

If the client does not receive a page, due to not all pages being sent, or packet loss (with the former having priority), it sends a packet with OPCODE 3, format :



where :

SERVERTOKEN is a copy of the SERVERTOKEN from the setup response.

COOKIE is a copy of COOKIE from the server response.

COUNT is the number of pages to be sent.

PAGE is the number of the first page to be sent.

PAGESIZE is a copy of PAGESIZE from the server response.

DATA is a copy of DATA from the initial request.

The server response is the same as in section 3.6.

Once a client has received all pages, it processes the complete assembled response as normal.

If the server encounters an error condition, such as an invalid SERVERTOKEN or COOKIE, it sends a setup response (section 3.3), and the client retries with a new initial request (section 3.4).

If a server has more than one IP address, a client MAY attempt to use a SERVERTOKEN it has previously acquired from another IP address. A client MAY also attempt to re-use a COOKIE to continue a failed transfer on an alternate server IP address or an alternative server.

3.8 Encryption and Authentication.

OPCODE 4 is used to optionally encrypt and authenticate packets using the algorithms described in [NACL]. Public keys are 255 bits. The wire format is 32 bytes, with the unused first bit set to zero. Public key tags are the first 4 bytes of the wire format public key.

For client requests, the OPCODE is followed by

o The 12 byte REQUESTID (which is the NACL client nonce).
o The 4 byte SERVERTOKEN.
o A 32 byte client public key.
o A 4 byte server public key tag.
o A cryptographic box containing a 16 byte MAC and the encrypted packet.

SERVERTOKEN is sent in the clear (and not in the encrypted packet) to allow servers to check client identity before performing public key operations. Setup packets (OPCODE 0) are not encrypted.

For server responses, the OPCODE is followed by

o The 12 byte REQUESTID (copied from the request).
o A 12 byte server nonce.
o A cryptographic box containing a 16 byte MAC and the encrypted packet.

In both cases the 12 byte REQUESTID is omitted from the unencrypted packet, which starts with the underlying OPCODE.

3.9 Congestion control

The number of pages requested but not received or lost (INFLIGHT) MUST be limited to a value (INFLIGHTMAX) so that undue network congestion

is avoided. Packets are deemed lost if they do not arrive within TIMEOUT milli-seconds of being requested.

For current DNS purposes (excluding AXFR) a simple method is to set INFLIGHTMAX = 4 and TIMEOUT = 1500 milli-seconds.

Alternatively, the following control algorithm MAY be used to allow higher performance. Set

INFLIGHTMAX = 4 + 3 * (RTT / PT) * (RTT / RTT_RECENT)

TIMEOUT = INFLIGHTHIGH * PT + 2 * RTT_MAX

where

RTT is the observed minimum round trip time based on a long sampling period.

PT is the smoothed observed time to transmit a full size packet based on a long sampling period.

RTT_RECENT is the smoothed observed round trip time, based on a short sampling period.

INFLIGHTHIGH is the highest value of INFLIGHT for the current transfer.

RTT_MAX is the maximum round trip time observed over a long sampling period.

The intention is that the the number of in-flight packets is quickly reduced in response to an increase in latency.

Sampling periods and smoothing filters need to be determined and tuned based on operational experience. "A long sampling period" might be the last 8 transfers. RTT_RECENT might be updated when a packet arrives by setting

RTT_RECENT = ALPHA * TRIP + (1-ALPHA) * RTT_RECENT

where

TRIP is the round trip time for the packet.

ALPHA is 0.1 if PACKETS > 10, otherwise 1.0 / PACKETS.

PACKETS is the number of packets received.

Other control algorithms MAY be employed, provided they do not cause a significant increase in latency (round trip time). Algorithms that increase INFLIGHT until packets are lost MUST NOT be used. Explicit Congestion Notification [RFC3168] MAY be used.

3.10 Status Codes

The following values are defined:

```
  0  No error
  1  Invalid SERVERTOKEN
  2  Invalid COOKIE

  11 Invalid OPCODE
  12 End of packet error
  13 Other format error

  31 Invalid PAGESIZE
  32 Invalid PAGE

  41 Invalid Public Key Tag
  42 Authentication error
```

Only codes 0-2 will occur if the protocol is correctly implemented, in the absence of network errors or attacks.

Servers MAY optionally generate status codes greater than 10. Such responses MAY be logged or used for debugging purposes, but MUST otherwise be ignored.

3.11 EDNS Tunnel

UDP over a port other than 53 is sometimes blocked by firewalls or network access gateways. In this case QRP queries and responses are sent by UDP/53 using an EDNS [RFC2671] option.

The DNS Message consists of a single OPT record in the additional section with an OPTION that carries the QRP message.

COUNT (the number of pages requested/sent) may need to be set to 1, since firewalls may prevent multiple responses being sent in response to a single query.

EDNS Tunnel is used if and only if the Port is 53.

3.12 Signalling

Clients discover QRP support and parameters by DS records [RFC4034] with Digest Type = QRP (201). The Key Tag is used to specify a Port number. The Algorithm field is set to NACL (204) or zero. The Digest is a list of Name / Key pairs : a lower case name server name relative to the zone followed by the 32-byte public key. If the Algorithm is zero, there is no public key, and encryption / authentication is not available.

For example:

```
example.com. 86400 DS (
  53    ; Port Number ( normally Key Tag )
  204   ; Algorithm = NACL
  201   ; Digest Type = QRP
  01 61 02 6E 73 00   ; Name server = a.ns.example.com.
  94B745D819AA0C50 3B2F06FC566250F4
  5E004F7D2BD69280 F96EC89E7FB40A6E ; 32 byte public key
  01 62 02 6E 73 00   ; Name server = b.ns.example.com.
  261EA433989353E9 1E987D5A3D3FB568
  BCC46A8CFCF25306 0AC4A9725E4E6F4C ; 32 byte public key
)
```

If the parent zone does not yet have full support for DS records, QRP parameters may instead be stored in the child zone using a CDS resource record, to allow partial "opportunistic" protection. The format of the CDS record is identical to a DS record. Authoritative servers should include the CDS RRset in the Authority section of an authoritative response when DO=1, so that subsequent queries to be performed using QRP.

4. Security Considerations

Fragmented responses are vulnerable to blind spoofing. If the path MTU is less than the value supplied by the client, denial of service

attacks are possible, and data can be altered unless authenticated by other means.

Amplification attacks from previous users of the client IP address on the current user are not prevented by the protocol until the long term server secret is changed, as described in section 3.3. In-path (man-in-the-middle) amplification attacks are not prevented, however such attacks are relatively difficult to carry out, requiring the attacker to have network access close to the victim.

Transactions not protected as described in section 3.8 are vulnerable to data alteration. Such attacks may be prevented by the use of DNSSEC.

Secret values need to be generated so that an attacker cannot easily guess them, by using cryptographic random number generators seeded from data that cannot be guessed by an attacker, such as thermal noise or other random physical fluctuations.

5. IANA Considerations

The following values may be used for private testing only :

```
QRP Digest type = 201
NACL Algorithm = 204
QRP Tunnel EDNS OPTION code = 200
CDS resource record type = 65351
```

IANA is requested to make official reservations, to allow public operation.

6. Acknowledgments

Mark Andrews, Alex Bligh, Matthew Dempsky, Robert Elz, Alfred Hoenes, Douglas Otis, Nicholas Weaver and Wouter Wijngaards were each instrumental in creating and refining this specification.

7. References

7.1 Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC768] J. Postel, "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, August 1980.

[NACL] Bernstein, D., "Cryptography in NaCl", April 2009.

[RFC3168] Ramakrishnan, K., "The Addition of Explicit Congestion Notification (ECN) to IP", September 2001.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for DNS Security Extensions", RFC 4034, March 2005.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

7.2 Informative References

[GONT] Gont, F., "Security Assessment of the Internet Protocol version 4", August 2009.

Appendix A. Implementation of Cookies

The suggested implementation of cookies is by version numbers. Each RRset has a version number assigned from a 64-bit clock that is increased whenever the DNS database is updated. The version of a response is the largest version number of the associated RRsets. The cookie is the version number.

If the database is updated while a transfer is progress, a COOKIE error occurs, and the client restarts the transfer.

Alternatively, if old queries may be replayed, COOKIE errors may be avoided(however such errors should be rare).

Appendix B. Anycast considerations

Anycast DNS servers need to operate consistently.
There are (at least) two possibilities:

(a) Each server within the Anycast system issues distinct SERVERTOKENS. If the Anycast routing changes, a SERVERTOKEN error occurs, and the client restarts the query.

(b) Each server within the Anycast system has the same long term secret, and thus issues the same SERVERTOKEN to a given client. A global clock is used for issuing updates. If the Anycast routing changes and an update is in progress, a COOKIE error may occur, and the client has to restart the query. Such errors can be avoided by not serving updates until all the Anycast servers have received a copy.

Author's Address

George Barwood
33 Sandpiper Close
Gloucester
GL2 4LZ
United Kingdom

Phone: +44 452 722670
EMail: george.barwood@blueyonder.co.uk