Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-hsyu-message-fragments-01
Published: 7 November 2022
Intended Status: Experimental
Expires: 11 May 2023
Authors: H. Yu Guangzhou Genlian
Y. Liu Guangzhou Genlian

DNS message fragments

Abstract

This document describes a method to transmit DNS messages over multiple UDP datagrams by fragmenting them at the application layer. The objective is to allow authoriative servers to successfully reply to DNS queries via UDP using multiple smaller datagrams, where larger datagrams may not pass through the network successfully.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Background

[RFC1035] describes how DNS messages are to be transmitted over UDP. A DNS query message is transmitted using one UDP datagram from client to server, and a corresponding DNS reply message is transmitted using one UDP datagram from server to client.

The upper limit on the size of a DNS message that can be transmitted thus depends on the maximum size of the UDP datagram that can be transmitted successfully from the sender to the receiver. Typically any size limit only matters for DNS replies, as DNS queries are usually small.

As a UDP datagram is transmitted in a single IP, in theory the size of a UDP datagram (including various lower internet layer headers) can be as large as 64 KiB. But practically, if the datagram size exceeds the path MTU, then the datagram will either be fragmented at the IP layer, or worse dropped, by a forwarder. In the case of IPv6, DNS packets are fragmented by the sender only. If a packet's size exceeds the path MTU, it must be fragmented. Except for the first fragmented package, other fragmented packages do not include a UDP or TCP header, and do not know the port number of the IP package, and the subsequent IP slice pack is filtered off. A Packet Too Big (PTB) ICMP message will be received by sender without any clue to the sender to reply again with a smaller sized message, due to the stateless feature of DNS. In addition, IP-level fragmentation caused by large DNS response packet will introduce risk of cache poisoning [Fragment-Poisonous], in which the attacker can circumvent some defense mechanisms (like port, IP, and query randomization [RFC5452]).

As a result, a practical DNS payload size limitation is necessary. [RFC1035] limited DNS message UDP datagram lengths to a maximum of 512 bytes. Although EDNS(0) [RFC6891] allows an initiator to advertise the capability of receiving lager packets (up to 4096 bytes), it leads to fragmentation because practically most packets are limited to 1500 byte size due to host Ethernet interfaces, or 1280 byte size due to minimum IPv6 MTU in the IPv6 stack [RFC3542].

According to DNS specifications [RFC1035], if the DNS response message can not fit within the packet's size limit, the response is truncated and the initiator will have to use TCP as a fallback to re-query to receive large response. However, not to mention the high setup cost introduced by TCP due to additional roundtrips, some firewalls and middle boxes even block TCP/53 which cause no responses to be received as well. It becomes a significant issue when the DNS response size inevitably increases with DNSSEC deployment.

In this memo, DNS message fragmentation attempts to work around middle box misbehavior by splitting a single DNS message across multiple UDP datagrams. Note that to avoid DNS amplification and reflection attacks, DNS cookies [I-D.ietf-dnsop-cookies] is a mandatory requirement when using DNS message fragments.

1.2. Motivation

It is not a new topic regarding large DNS packets(>512B) issue [I-D.ietf-dnsop-respsize], starting from introduction of IPv6, EDNS(0) [SAC016], and DNSSEC deployment [SAC035]. In current production networks, using DNSSEC with longer DNSKEYs (ZSK>1024B and KSK>2048B) will result in response packets no smaller than 1500B [T-DNS]. Especially during the KSK rollover process, responses to the query of DNSKEY RRset will be enlarged as they contain both the new and old KSK.

When possible, we should avoid dropped packets as this means the client must wait for a timeout, which incurs a high cost. For example, a validator behind a firewall suffers waiting till the timeout with no response, if the firewall drops large EDNS(0) packets and IP fragments. It may even cause disaster when the validator can not recieve response for new trust anchor KSK due to the extreme case of bad middle boxes which also drop TCP/53.

Since UDP requires fewer packets on the wire and less state on servers than TCP, in this memo we propose continuing to use UDP for transmission but fragment the larger DNS packets into smaller DNS packets at the application layer. We would like the fragments to easily go through middle boxes and avoid falling back to TCP.

2. DNS Message Fragmentation Method

2.1. Client Behavior

Clients supporting DNS message fragmentation add an EDNS option to their queries, which declares their support for this feature.

If a DNS reply is received that has been fragmented, it will consist of multiple DNS message fragments (each transmitted in a respective UDP packet), and every fragment contain an EDNS option which says how many total fragments there are, and the identifier of the fragment that the current packet represents. The client collects all of the fragments and uses them to reconstruct the full DNS message. Clients MUST maintain a timeout when waiting for the fragments to arrive.

Clients that support DNS message fragments MUST be able to reassemble fragments into a DNS message of any size, up to the maximum of 64KiB.

The client MAY save information about what sizes of packets have been received from a given server. If saved, this information MUST have a limited duration.

Any DNSSEC validation is performed on the reassembled DNS message.

2.2. Server Behavior

Servers supporting DNS message fragmentation will look for the EDNS option which declares client support for the feature. If not present, the server MUST NOT use DNS message fragmentation. The server MUST check that DNS cookies are supported. [**FIXME**] Implementation of the first request case, where no existing established cookie is available needs discussion; we want to avoid additional round-trips here. Shane: don't cookies already handle this case?

The server prepares the response DNS message normally. If the message exceeds the maximum UDP payload size specified by the client, then it should fragment the message into multiple UDP datagrams.

Each fragment contains an identical DNS header with TC=1, possibly varying only in the section counts. Setting the TC flag in this way insures that clients which do not support DNS fragments can fallback to TCP transparently.

As many RR are included in each fragment as are possible without going over the desired size of the fragment. An EDNS option is added to every fragment, that includes both the fragment identifier and the total number of fragments.

The server needs to know how many total fragments there are to insert into each fragment. A simple approach would be to generate all fragments, and then count the total number at the end, and update the previously-generated fragments with the total number of fragments. Other techniques may be possible.

The server MUST limit the number of fragments that it uses in a reply. (See "Open Issues and Discussion" for remaining work.)

The server MUST NOT exceed the maximum fragment size requested by a client.

The server should use the following sizes for each fragment in the sequence in IPv4:

Fragment ID	Size
1	min(512, client_specified_max)
2	min(1460, client_specified_max)
3	min(1480, client_specified_max)
N	min(1480, client_specified_max)

Table 1

The rationale is that the first packet will always get through, since if a 512 octet packet doesn't work, DNS cannot function. We then increase to sizes that are likely to get through. 1460 is the 1500 octet Ethernet packet size, minus the IP header overhead and enough space to support tunneled traffic. 1480 is the 1500 octet Ethernet packet size, minus the IP header overhead. [**FIXME**] Why not add 1240 here? Shane answers: 1280 is not any kind of limit in IPv4, as far as I know.

The server should use the following sizes for each packet in the sequence in IPv6:

Fragment ID	Size
1	min(1240, client_specified_max)
2	min(1420, client_specified_max)
3	min(1460, client_specified_max)
N	min(1460, client_specified_max)

Table 2

Like with IPv4, the idea is that the first packet will always get through. In this case we use the IPv6-mandated 1280 octets, minus the IP header overhead. We then increase to 1420, which is the 1500 octet Ethernet packet size, minus the IP header overhead and enough space to support tunneled traffic. 1460 is the 1500 octet Ethernet packet size, minus the IP header overhead.

2.3. Other Notes

- The FRAGMENT option MUST NOT be present in DNS query messages, i.e., when QR=0. If a DNS implementation notices the FRAGMENT option in a DNS query message, it MUST ignore it.

- In DNS reply messages, the FRAGMENT option MUST NOT be present in datagrams when truncation is not done, i.e., when TC=0. If a DNS implementation notices the FRAGMENT option in a DNS reply message fragment datagram that is not truncated, i.e, when TC=0, it MUST drop all DNS reply message fragment datagrams received so far (awaiting assembly) for that message's corresponding question tuple (server IP, port, message ID) without using any data from them. [**FIXME**] Dropping fragments to be received yet will be problematic for implementations, but dropping fragments received so far ought to be sufficient.

- More than one FRAGMENT option MUST NOT be present in a DNS reply message fragment datagram. If a DNS implementation notices multiple FRAGMENT options in a DNS reply message fragment datagram, it MUST drop all reply datagrams received for that message's corresponding question tuple (server IP, port, message ID) without using any data from them. [**FIXME**] Dropping fragments to be received yet will be problematic for implementations, but dropping fragments received so far ought to be sufficient.

3. The ALLOW-FRAGMENTS EDNS(0) Option

ALLOW-FRAGMENTS is an EDNS(0) [RFC6891] option that a client uses to inform a server that it supports fragmented responses. [**FIXME**] Why not simply use the FRAGMENT option here with count=0, identifier=ignored and avoid using another option code? Shane: There are no shortage of options. Plus, if we want to include a maximum fragment size value in the ALLOW-FRAGMENTS then we really need a separate option.

3.1. Wire Format

TBD.

3.2. Option Fields

3.2.1. Maximum Fragment Size

The Maximum Fragment Size field is represented as an unsigned 16-bit integer. This is the maximum size used by any given fragment the server returns. [**FIXME**] This field's purpose has to be explained. Shane: discussed in the discussion section now.

3.3. Presentation Format

As with other EDNS(0) options, the ALLOW-FRAGMENTS option does not have a presentation format.

4. The FRAGMENT EDNS(0) Option

FRAGMENT is an EDNS(0) [RFC6891] option that assists a client in gathering the various fragments of a DNS message from multiple UDP datagrams. It is described in a previous section. Here, its syntax is provided.

4.1. Wire Format

TBD.

4.2. Option Fields

4.2.1. Fragment Identifier

The Fragment Identifier field is represented as an unsigned 8-bit integer. The first fragment is identified as 1. Values in the range [1,255] can be used to identify the various fragments. Value 0 is used for signalling purposes.

4.2.2. Fragment Count

The Fragment Count field is represented as an unsigned 8-bit integer. It contains the number of fragments in the range [1,255] that make up the DNS message. Value 0 is used for signalling purposes.

4.3. Presentation Format

As with other EDNS(0) options, the FRAGMENT option does not have a presentation format.

5. Network Considerations

5.1. Background

TCP-based application protocols co-exist well with competing traffic flows in the internet due to congestion control methods such as in [RFC5681] that are present in TCP implementations.

UDP-based application protocols have no restrictions in lower layers to stop them from flooding datagrams into a network and causing congestion. So applications that use UDP have to check themselves from causing congestion so that their traffic is not disruptive.

In the case of [RFC1035], only one reply UDP datagram was sent per request UDP datagram, and so the lock-step flow control automatically ensured that UDP DNS traffic didn't lead to congestion. When DNS clients didn't hear back from the server, and had to retransmit the question, they typically paced themselves by using methods such as a retransmission timer based on a smoothed round-trip time between client and server.

Due to the message fragmentation described in this document, when a DNS query causes multiple DNS reply datagrams to be sent back to the client, there is a risk that without effective control of flow, DNS traffic could cause problems to competing flows along the network path.

Because UDP does not guarantee delivery of datagrams, there is a possibility that one or more fragments of a DNS message will be lost during transfer. This is especially a problem on some wireless networks where a rate of datagrams can continually be lost due to interference and other environmental factors. With larger numbers of message fragments, the probability of fragment loss increases.

5.2. Implementation Requirements

TBD.

6. Open Issues and Discussion

1. Resolver behavior

   We need some more discussion of resolver behavior in general, at least to the point of making things clear to an implementor.

2. The use of DNS fragments mechanism

   Is this mechanism designed for all DNS transactions, or only used in some event or special cases like a key rollover process? If the mechanism is designed for general DNS transactions, when is it triggered and how is it integrated with existing patterns?

   One option is that DNS fragments mechanism works as a backup with EDNS, and triggered only when a larger packet fails in the middle. It will be orthogonal with TCP which provide additional context that TC bit will be used in server side.

3. What is the size of fragments?

   Generally speaking the number of fragment increases if fragment size is small (512 bytes, or other empirical value), which makes the mechanism less efficient. If the size can changed dynamically according to negotiation or some detection, it will introduce more cost and round trip time.

4. What happens if a client that does not support DNS fragments receives an out-of-order or partial fragment?

   We need to consider what happens when a client that does not support DNS fragments gets a partial response, possibly even out of order.

5. We should explain risk of congestion, packet loss, etc. when introducing the limit on the number of fragments. We might also set specific upper limits for number of fragments.

6. EDNS buffer sizes vs. maximum fragmentation sizes

   Mukund Sivaraman: We need further discussion about the sizes; also an upper limit for each *fragment* has to be the client's UDP payload size as it is the driver and it alone knows the ultimate success/failure of message delivery. So if it sets a maximum payload size of 1200, there's no point in trying 1460. Clients that support DNS message fragments (and signal support using the EDNS option) should adapt their UDP payload size discovery algorithm to work with this feature, as the following splits on sizes will assist PMTU discovery.

   Shane Kerr: I think we need to separate the EDNS maximum UDP payload size from the maximum fragment size. I think that it is quite likely that (for example) we will want to restrict each fragment to 1480 bytes, but that the EDNS buffer size might remain at 4 kibibytes.

7. TSIG should be addressed

   We need to document how to handle TSIG, even though this is not likely to be a real-world issue. Probably each fragment should be TSIG signed, as this makes it harder for an attacker to inject bogus packets that a client will have to process.

8. RR splitting should be addressed

   We need to document whether or not RR can be split. Probably it makes sense not to allow this, although this will reduce the effectiveness of the fragmentation, as the units that can be packed into each fragment will be bigger.

9. We need to document that some messages may not be possible to split.

   Some messages may be too large to split. A trivial example is a TXT record that is larger than the buffer size. Probably the best behavior here is to truncate.

10. DNSSEC checks

    DNSSEC checks should be done on the final reassembled packet. This needs to be documented.

11. Name compression

    Name compression should be done on the each fragment separately. This needs to be documented.

12. OPT-RR

    Some OPT-RR seem to be oriented at the entire message, others make more sense per packet. This needs to be sorted out. Also we need to investigate the edge case where fragments have conflicting options (Mukund Sivaraman thinks that we can copy the approach in the EDNS specification and use the same rules about conflicting OPT-RR that it uses.)

7. Security Considerations

To avoid DNS amplification or reflection attacks, DNS cookies [I-D.ietf-dnsop-cookies] must be used. The DNS cookie EDNS option is identical in all fragments that make up a DNS message. The duplication of the same cookie values in all fragments that make up the message is not expected to introduce a security weakness in the case of off-path attacks.

8. IANA Considerations

The ALLOW-FRAGMENTS and FRAGMENT EDNS(0) options require option codes to be assigned for them.

9. Acknowledgements

Thanks to Stephen Morris, JINMEI Tatuya, Paul Vixie, Mark Andrews, and David Dragon for reviewing a pre-draft proposal and providing support, comments and suggestions.

10. References

[Fragment-Poisonous] Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", 2012.

[I-D.ietf-dnsop-cookies] Eastlake, D. E. and M. P. Andrews, "Domain Name System (DNS) Cookies", Work in Progress, Internet-Draft, draft-ietf-dnsop-cookies-04, 1 July 2015, <https://www.ietf.org/archive/id/draft-ietf-dnsop-cookies-04.txt>.

[I-D.ietf-dnsop-respsize] Vixie, P., Kato, A., and J. Abley, "DNS Referral Response Size Issues", Work in Progress, Internet-Draft, draft-ietf-dnsop-respsize-15, 14 February 2014, <https://www.ietf.org/archive/id/draft-ietf-dnsop-respsize-15.txt>.

[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>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <https://www.rfc-editor.org/info/rfc3542>.

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <https://www.rfc-editor.org/info/rfc5452>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6891]
Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[SAC016] ICANN Security and Stability Advisory Committee, "Testing Firewalls for IPv6 and EDNS0 Support", 2007.

[SAC035] ICANN Security and Stability Advisory Committee, "DNSSEC Impact on Broadband Routers and Firewalls", 2008.

[T-DNS] Zhu, L., Hu, Z., and J. Heidemann, "T-DNS: Connection-Oriented DNS to Improve Privacy and Security (extended)", 2007, <http://www.isi.edu/~johnh/PAPERS/Zhu14b.pdf>.

Authors' Addresses

Haisheng Yu
Guangzhou Genlian
Xiangjiang International Technology Innovation Center, 41 Jinlong Road, Nansha District, Guangzhou
Guangzhou
China

Email: hsyu@cfiec.net

Yan Liu
Guangzhou Genlian
Xiangjiang International Technology Innovation Center, 41 Jinlong Road, Nansha District, Guangzhou
Guangzhou
China

Email: yliu@cfiec.net