Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: March 31, 2016

M. Sivaraman
Internet Systems Consortium
September 28, 2015

DNS message checksums
draft-muks-dnsop-dns-message-checksums-00

Abstract

This document describes a method for a client to be able to verify that IP-layer PDU fragments of a UDP DNS message have not been spoofed by an off-path attacker.

Table of Contents

1. Introduction

[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.

As a UDP datagram is transmitted in a single IP PDU, 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 dropped by a forwarder. In the case of IPv4, DNS datagrams may be fragmented by a sender or a forwarder. In the case of IPv6, DNS datagrams are fragmented by the sender only.

IP-layer fragmentation for large DNS response datagrams introduce risk of cache poisoning by off-path attackers [Fragment-Poisonous] in which an attacker can circumvent some defense mechanisms like port, IP, and query randomization [RFC5452].

This memo introduces the concept of a DNS message checksum which may be used to stop the effects of such off-path attacks.

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].

2. DNS message checksum method

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

The CHECKSUM EDNS option contains 3 fields: NONCE, ALGORITHM, and DIGEST. These fields are described in Section 3.

It is OPTIONAL for a client to add a CHECKSUM EDNS option to DNS query messages. If it adds such an option, it MUST set the NONCE field to a random 128-bit unsigned integer. The ALGORITHM field MUST be set to 0 and the DIGEST field MUST be left empty. The NONCE field MUST be randomly generated (i.e., in no predictable sequence) for each query for which the client uses a CHECKSUM EDNS option. The client is expected to remember the per-query NONCE field's value to be used in verifying the reply to this query message.

A client MUST NOT send multiple DNS query messages with the NONCE set to a fixed unchanging value. Instead, it must not send the option at all.

The server SHOULD add a CHECKSUM EDNS option in the reply message to a corresponding query that arrived with this option present. The NONCE field MUST be copied verbatim from the query message to the corresponding reply message. A checksum is computed over the DNS reply message as described in Section 4 and the ALGORITHM and DIGEST fields MUST be set using the resulting checksum as described in Section 3. The server is at liberty to choose any checksum algorithm it wants to. A list of algorithms is given in Appendix A.

When a client receives a reply message for which it sent a CHECKSUM EDNS option in the corresponding query, it SHOULD look for the presence of the CHECKSUM EDNS option in the reply. The client may handle the lack of a CHECKSUM EDNS option in the reply as it chooses to.

If a CHECKSUM EDNS option is present in the reply, the client SHOULD first check and ensure that the NONCE field contains the same nonce value that was sent in the corresponding query message. If the value in the NONCE field is different, the reply message MUST be discarded. Afterwards, the client SHOULD proceed to compute a checksum over the reply message as described in Section 4 using the checksum algorithm in the ALGORITHM field. It SHOULD then compare the checksum value with the value that was received in the DIGEST field for equality. If they are not equal, the reply message MUST be discarded. If they are equal, the reply message can be used normally as the client intends to use it.

3. The CHECKSUM EDNS(0) option

CHECKSUM is an EDNS(0) [RFC6891] option that is used to transmit a digest of a DNS message in replies. Its use is described in a previous section. Here, its syntax is provided.

3.1. Wire format

The following describes the wire format of the OPTION-DATA field [RFC6891] of the CHECKSUM EDNS option. All CHECKSUM option fields must be represented in network byte order.

Option field	Type	Field size
NONCE	unsigned integer	128 bits (16 octets)
ALGORITHM	unsigned integer	16 bits (2 octets)
DIGEST	byte array	Variable length

3.2. Option fields

3.2.1. NONCE

The NONCE field is represented as an unsigned 128-bit integer in network byte order. It MUST be randomly computed for each query message which a client sends out, and is copied verbatim from the query to the corresponding reply DNS message by the server.

3.2.2. ALGORITHM

The ALGORITHM field is represented as an unsigned 16-bit integer in network byte order. In query messages, it MUST be set to 0. In reply messages, it MUST contain the numeric value of the algorithm used to compute the DIGEST field. A list of algorithms and their values is given in Appendix A.

3.2.3. DIGEST

The DIGEST field is represented as a sequence of octets present after the NONCE and ALGORITHM fields. Its size is implicitly computed from the value in the OPTION-LENGTH field [RFC6891] for the CHECKSUM EDNS option minus the size of the NONCE and ALGORITHM fields. In query messages, it MUST be empty. In reply messages, it MUST contain the digest of the reply message which is computed as described in Section 4.

3.3. Presentation format

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

4. Checksum computation

To generate the checksum digest to be placed in the DIGEST field, first the entire DNS message must be prepared (rendered) along with the CHECKSUM option embedded in it to the point that it is ready to be sent out on the wire. In this CHECKSUM option, initially the DIGEST field must be filled with zero values and its size must be reserved equal to the size expected for the digest from the checksum algorithm intended to be used. The NONCE field MUST be set to the value of the nonce from the query DNS message. The ALGORITHM field MUST be set to the checksum algorithm intended to be used. After this, the whole message contents (from the start of the DNS message header onwards) must be input to the checksum algorithm and the calculated checksum must be patched into the DIGEST field, space for which was reserved before.

To verify the checksum digest from a DNS message that was received, first the DIGEST field is copied to a temporary location and the DIGEST field in the message is patched with zero values. After this, the whole message contents (from the start of the DNS message header onwards) must be input to the checksum algorithm specified in the ALGORITHM field. The calculated checksum must be compared for equality with the checksum originally received in the DIGEST field, the content of which was earlier saved to a temporary location. If both are equal, the checksum matches.

5. Security considerations

The methods in this memo are designed to thwart off-path spoofing attacks which may lead to cache-poisoning, including the specific case when IP-layer PDU fragmentation occurs.

The CHECKSUM EDNS option is not designed to offer any protection against on-path attackers. Very little can be done without using strong cryptographic methods for this case.

Checksum computation may increase resource usage on servers and clients. It is thus desirable to use fast checksum algorithms which provide ample security to verify a short-lived DNS message.

The entropy source used for generating random values for use in the NONCE field may be chosen similarly to provide ample security to verify a short-lived DNS message.

The NONCE field effectively extends the ID field [RFC1035] in the DNS message header.

As a side-effect of using checksums, resolver cache poisoning attacks are made more difficult due to the presence of the NONCE field.

The CHECKSUM EDNS option cannot prevent some kinds of attack such as response and NS blocking and NS pinning as described in [Fragment-Poisonous].

6. IANA Considerations

The CHECKSUM EDNS(0) option requires an option code to be assigned for it. Checksum algorithms in Appendix A need to be registered as well.

7. Acknowledgements

Thanks to Tomek Mrugalski for offering tips on draft naming and upload process.

8. References

8.1. 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>.

[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>.

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

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

8.2. Informative references

[Fragment-Poisonous]

Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", 2012.

Appendix A. Checksum algorithms

The ALGORITHM field identifies the checksum algorithm that is used to compute the checksum digest for a DNS message. Fast checksum algorithms which are able to provide ample security to verify a short-lived DNS message are sufficient.

The following table lists the currently defined checksum algorithm types.

Value	Type	Status, Remarks
0	EMPTY	Empty digest
1	SHA-1	Mandatory

Appendix B. Change History (to be removed before publication)

- o draft-muks-dnsop-dns-message-checksums-00
  Initial draft (renamed version). Removed the NONCE-COPY field as it is no longer necessary. Doubled the size of the NONCE field to 128 bits. Added sample checksum algorithms. Fixed incorrect reference, language and grammar.

Author's Address

Mukund Sivaraman
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: muks@isc.org
URI: http://www.isc.org/