dnsop
Internet-Draft
Intended status: Standards Track
Expires: 15 July 2026

X.L. Li
Y.Q. Qiu
Nankai University

11 January 2026

DNS Response Pre-processing Security Guidelines: Awaiting Valid Responses
draft-li-dnsop-response-preprocessing-01

Abstract

The security and robustness of the Domain Name System (DNS) significantly depend on how resolvers handle received responses. Current DNS specifications lack exhaustive and consistent guidance on response pre-processing, particularly for malformed or unexpected packets. This specification gap has led to implementation divergences and has been shown to introduce security vulnerabilities such as DNS cache poisoning, Denial of Service (DoS), and resource exhaustion, as detailed in the TUDOOR attack research.

This document aims to clarify and standardize the behavior of DNS resolvers when receiving and initially processing responses from upstream servers. The core proposal is the "Awaiting Valid Responses" mechanism, which mandates that a resolver, after issuing a query, MUST maintain a defined waiting period to receive a well-formed, relevant, and validated response. During this period, non-compliant incoming packets should be discarded, and the resolver should continue to wait until a valid response is received or the query times out. This document provides guidance for DNS implementers to mitigate security risks arising from flaws in response pre-processing logic.

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 July 2026.

Table of Contents

1. Introduction 3
1.1. Background 3
1.2. Problem Statement 3
1.3. Document Goals and Scope 3
2. Requirements Language 4
3. Terminology 4
4. The "Awaiting Valid Responses" Mechanism 5
4.1. Core Principles 5
4.2. Recommended State Model for Response Pre-processing . . . 6
4.2.1. State 0: Awaiting Response 6
4.2.2. State 1: Reception and Network Layer Validation . . . 6
4.2.3. State 2: ICMP Message Evaluation 7
4.2.4. State 3: Transport Layer Validation 7
4.2.5. State 4: DNS Header Validation 8
4.2.6. State 5: DNS Section Parsing and Validation 8
4.2.7. State 6: Transaction ID (TXID) Validation 9
4.2.8. State 7: Valid Response Received 9
4.2.9. State 8: Query Timeout 9
4.2.10. State 9: Retransmission Decision 10
4.2.11. State 10: Resolution Termination 10
4.3. General Guidelines for Handling Malformed and Unexpected Packets 10
5. Considerations for Known Attack Vectors 11
5.1. Regarding Cache Poisoning (V_CP) 11
5.2. Regarding Denial of Service (V_DS) 11
5.3. Regarding Resource Exhaustion (V_RC) 11
6. Security Considerations 11
7. Relationship to Other RFCs 12
8. IANA Considerations 12
9. References 12

9.1. Normative References 13
9.2. Informative References 13
Authors' Addresses 15

1. Introduction

1.1. Background

The Domain Name System (DNS) [RFC1034], [RFC1035] is a fundamental component of the Internet's infrastructure. DNS resolvers, including recursive resolvers, forwarders, and stub resolvers, interact with upstream DNS servers and process their responses in the course of resolving domain names to IP addresses. Response pre-processing is a critical phase in this interaction, involving initial checks and validations of incoming packets before the formal processing of DNS record content. Additional clarifications on DNS behavior can be found in [RFC2181] and [RFC2308].

1.2. Problem Statement

While the basic DNS protocol is defined in [RFC1034] and [RFC1035], these specifications do not provide sufficient detail on how resolvers should handle various forms of invalid, malformed, or unexpected response packets. For instance, Section 4.1.2 of [RFC1035] describes header fields but offers limited guidance on how a receiver should validate and react to non-compliant header fields. This lack of specificity has led to diversity in DNS software implementations.

Research [TUDOOR] has shown that these implementation differences and improper handling of edge cases can create exploitable logical vulnerabilities. Attackers can trigger these vulnerabilities by sending specially crafted malformed DNS responses, leading to consequences such as DNS cache poisoning (see also [KAMINSKY08], [SAD-DNS], and [FRAG-POISON]), Denial of Service (DoS), or resolver resource exhaustion. These attacks leverage unexpected resolver behaviors during the response pre-processing phase, such as premature termination of resolution, erroneous acceptance of invalid data, or bypassing internal control mechanisms.

1.3. Document Goals and Scope

The primary goal of this document is to provide clear guidance and best practices for the response pre-processing logic of DNS resolvers (including recursive resolvers, forwarders, and stub resolvers). Central to this is the introduction and formalization of the "Awaiting Valid Responses" mechanism.

This document aims to:

* Clarify the sequence of validation steps and behaviors that resolvers should perform upon receiving packets from upstream servers.

* Emphasize the principle of not prematurely terminating a resolution attempt upon receipt of an invalid or malformed packet.

* Provide recommendations for handling various common scenarios involving malformed and unexpected packets.

* Reduce the occurrence of logical vulnerabilities by standardizing pre-processing behavior, thereby enhancing the overall security and robustness of the DNS infrastructure.

This document does not attempt to modify the core data formats of the DNS protocol or the fundamental flow of query/response exchanges. Instead, it focuses on clarifying and supplementing existing specifications with respect to response reception and initial validation.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

Resolver: An entity that performs DNS query resolution. This can be a recursive resolver, a DNS forwarder, or a stub resolver.

Upstream Server: A server to which a resolver sends DNS queries, e.g., an authoritative name server or another recursive resolver.

Response Pre-processing: The set of checks and validation steps performed by a resolver upon receiving a packet from an upstream server, prior to the formal processing of DNS data content (e.g., caching records, returning to a client).

Malformed Packet: A packet that does not conform to the format requirements of the DNS protocol specification [RFC1035] or relevant network protocols (IP, UDP, TCP).

Unexpected Packet: A packet that, while potentially well-formed,

does not match the resolver’s current resolution context in terms of its content (e.g., mismatched TXID, incorrect QR bit) or timing.

Valid Response: A well-formed response packet whose content (especially the TXID) matches an active query and which adheres to DNS protocol logic.

Awaiting Valid Response Window: The period during which a resolver, after sending a query, waits to receive a valid response, typically limited by a query timeout mechanism.

Four-tuple: The combination of IP layer and transport layer identifiers, typically including source IP address, destination IP address, source port number, and destination port number.

Transaction ID (TXID): A 16-bit identifier in the DNS header used to match queries and responses.

TUDOOR Attack: A class of attacks described in the [TUDOOR] paper that exploit logical vulnerabilities in DNS response pre-processing.

4. The "Awaiting Valid Responses" Mechanism

4.1. Core Principles

A DNS resolver, after sending a query to an upstream server, MUST initiate an internal "Awaiting Valid Responses" cycle. During this cycle, the resolver’s primary objective is to receive and validate a valid response corresponding to the query sent.

The following principles MUST be adhered to:

* Persistent Waiting: After issuing a query, a resolver MUST continuously listen for and process incoming packets until either a response confirmed as valid is received, or the timeout mechanism for that query is triggered.

* Strict Validation: All received packets MUST undergo a series of predefined validation steps (detailed in Section 4.2) before being accepted as a valid response.

* Preferential Discarding of Invalid Packets: Any packet identified during validation as malformed, unexpected, or unrelated to an active query MUST be discarded. After discarding, the resolver MUST continue to await subsequent, potentially valid responses and SHOULD NOT terminate the resolution attempt for the original query

solely due to the receipt of an invalid packet. An exception is if a packet (e.g., a trusted ICMP message) definitively indicates a permanent failure for that query.

* Avoidance of Premature Termination: A resolver MUST NOT prematurely terminate the resolution process for an original query or close the associated network socket due to the receipt of most types of malformed or unexpected packets.

* Timeout Handling: If a valid response is not received within the pre-set Awaiting Valid Response Window, the resolver MUST follow its established timeout and retransmission logic.

4.2. Recommended State Model for Response Pre-processing

The following state model describes the recommended response pre-processing flow. A resolver maintains an independent state for each active outbound query.

4.2.1. State 0: Awaiting Response

Description: The resolver has sent a query upstream and is awaiting a response. A timeout timer is active.

Entry Condition: After sending a query.

Action: Listen on the network socket.

4.2.2. State 1: Reception and Network Layer Validation

Description: An IP packet is received from the network interface.

Actions:

1. Validate the packet’s network layer header (e.g., IP [RFC0791] header checksum). If invalid, the packet MUST be discarded.

2. Check if the packet’s four-tuple matches the expected response source for any active query. If no match, the packet MUST be discarded.

3. If passed, proceed to the next state based on protocol type (e.g., ICMP, UDP, TCP). For UDP/TCP, proceed to State 3 (Section 4.2.4). For ICMP, proceed to State 2 (Section 4.2.3).

4.2.3. State 2: ICMP Message Evaluation

Description: The received packet is an ICMP [RFC0792][RFC4443] message.

Actions:

1. The resolver SHOULD inspect the ICMP message type, code, and the quoted original datagram information in its payload to determine if it is relevant and trustworthy for an active query.

2. For ICMP messages indicating Port Unreachable (Type 3, Code 3) that match an active query's authoritative server and port, the resolver MAY consider this a failure signal for that specific upstream server/port combination and act accordingly (e.g., try an alternate server, or mark this query attempt as failed after multiple tries).

3. For other types of ICMP errors, especially those of unknown origin or ambiguous meaning, the resolver SHOULD NOT terminate the entire resolution process solely based on this ICMP message. It is RECOMMENDED to log the message and return to State 0 (Section 4.2.1) to continue waiting.

4. The resolver MUST NOT prematurely terminate resolution due to specific ICMP message combinations that can be exploited for DoS, as described in [TUDOOR] and [RFC5927].

4.2.4. State 3: Transport Layer Validation

Description: The received packet is a UDP [RFC0768] or TCP [RFC9293] segment.

Actions:

1. Validate the transport layer header (e.g., UDP/TCP checksum, if applicable and checked). If invalid, the packet MUST be discarded.

2. Check the UDP/TCP payload length. If the payload is empty or its length is insufficient to contain a minimal DNS header (12 bytes), the packet MUST be discarded. The resolver returns to State 0 (Section 4.2.1).

3. If the payload length is sufficient, proceed to State 4 (Section 4.2.5).

4.2.5. State 4: DNS Header Validation

Description: Validation of the DNS message header fields.

Actions:

1. Parse the DNS header.

2. QR Bit: MUST be 1 (response). If the QR bit is 0 (query), this packet MUST be discarded. The resolver returns to State 0 (Section 4.2.1). This is intended to prevent V_CP vulnerabilities like the one affecting Microsoft DNS in [TUDOOR].

3. OpCode: Should typically be 0 (standard query). Responses to other OpCodes should be handled according to their definitions and local policy. If the OpCode is invalid or unsupported, the packet SHOULD be discarded.

4. QDCOUNT: For a response, this field should generally match the QDCOUNT in the corresponding query. In many cases, it should be 1. If QDCOUNT does not meet expectations (e.g., is much larger than 1, or inconsistent with the query), the packet SHOULD be treated as suspicious and potentially discarded, depending on local policy and strict interpretation of [RFC1035]. Refer to [draft-ietf-dnsop-qdcount-is-one].

5. Other header fields (e.g., AA, TC, RD, RA, Z, RCODE) MUST be checked for validity within a response context. Headers with overtly contradictory or illegal values MUST cause the packet to be discarded. The resolver returns to State 0 (Section 4.2.1).

6. If header fields are preliminarily valid, proceed to State 5 (Section 4.2.6).

4.2.6. State 5: DNS Section Parsing and Validation

Description: Parsing and validation of the Question, Answer, Authority, and Additional sections of the DNS message body.

Actions:

1. Parse the sections according to the counts in the header.

2. Validate domain name representations (including the validity of compression pointers), RR types, and RDATA formats.

3. Any critical error that prevents unambiguous parsing of record content or violates specified formats (e.g., invalid compression pointer loops, RDLENGTH mismatch with actual data) MUST cause the packet to be discarded. The resolver returns to State 0 (Section 4.2.1).

4. If all sections are parsed successfully, proceed to State 6 (Section 4.2.7).

4.2.7. State 6: Transaction ID (TXID) Validation

Description: Validation that the TXID in the response matches that of an active query.

Actions:

1. If the response TXID matches an active query’s TXID, proceed to State 7 (Section 4.2.8).

2. If the TXID does not match, this packet MUST be discarded. The resolver returns to State 0 (Section 4.2.1).

4.2.8. State 7: Valid Response Received

Description: A valid response, having passed all pre-processing validations, has been received.

Actions:

1. Stop the timeout timer associated with this query.

2. Pass the validated response data to subsequent DNS processing modules (e.g., caching, responding to the client).

3. The pre-processing flow for this query concludes.

4.2.9. State 8: Query Timeout

Description: The timeout timer for awaiting a valid response has expired.

Actions:

1. If retransmission is permitted and the maximum retransmission count has not been reached, proceed to State 9 (Section 4.2.10).

2. Otherwise, proceed to State 10 (Section 4.2.11).

4.2.10. State 9: Retransmission Decision

Description: Deciding whether to retransmit the query.

Actions:

1. The resolver MUST check if the configured retransmission limit for this query has been reached. This check MUST occur before an actual retransmission is performed, to prevent V_RC vulnerabilities as described in [TUDOOR].

2. If the limit has not been reached, the resolver MAY retransmit the query (potentially with a new TXID and/or source port). After retransmission, return to State 0 (Section 4.2.1).

3. If the limit has been reached, proceed to State 10 (Section 4.2.11).

4.2.11. State 10: Resolution Termination

Description: The resolution attempt for this query has terminated.

Actions:

1. Return an appropriate error response (e.g., SERVFAIL) to the original requestor.

2. Clean up state associated with this query.

4.3. General Guidelines for Handling Malformed and Unexpected Packets

* Silent discard: For most packets determined to be invalid during pre-processing, the resolver SHOULD discard them silently, i.e., without sending any form of response to the source. This avoids leaking information about the resolver’s internal state or behavior.

* Persistent Listening: After discarding an invalid packet, the resolver MUST continue to listen on the socket for other potential responses related to the original query, until the query times out.

* Logging: A resolver MAY log diagnostic information about discarded malformed or unexpected packets. However, the level and frequency of logging should be configurable to prevent the logging system itself from becoming an attack vector or performance bottleneck.

* No State Contamination: The processing of invalid packets MUST NOT negatively affect the state of the resolver's other active queries or its shared cache.

5. Considerations for Known Attack Vectors

The specifications in this document are intended to mitigate security vulnerabilities arising from flaws in response pre-processing logic, particularly those highlighted in the [TUDOOR] research.

5.1. Regarding Cache Poisoning (V_CP)

By mandating in State 4.2.5 (Section 4.2.5) that the QR bit in a response MUST be 1 and that packets with QR=0 are discarded, attacks that exploit response sockets to probe source ports or inject queries are prevented. Concurrently, strict TXID matching in State 4.2.7 (Section 4.2.7) is fundamental to preventing cache poisoning.

5.2. Regarding Denial of Service (V_DS)

By emphasizing that the resolver MUST continue to await a valid response (return to State 0 (Section 4.2.1)) after receiving most types of malformed packets (including specific ICMP messages, empty-payload UDP packets, etc.), rather than prematurely terminating resolution (as specified in States 4.2.3 (Section 4.2.3), 4.2.4 (Section 4.2.4)), the effectiveness of such DoS attacks is significantly reduced. Failure should only be determined after a genuine query timeout or exhaustion of retransmission limits.

5.3. Regarding Resource Exhaustion (V_RC)

By explicitly requiring in State 4.2.10 (Section 4.2.10) that the retransmission count limit MUST be checked before any retransmission decision, attacks that induce a resolver to send an excessive number of queries to exhaust its resources are prevented.

6. Security Considerations

The guidelines in this document aim to enhance the resilience of DNS resolvers against specific types of attacks. However, implementers should be aware of the following:

* Implementation Correctness: The effectiveness of these guidelines depends on their correct and complete implementation. Any deviation from the state machine logic could reintroduce vulnerabilities.

* Resource Limits: While handling high volumes of incoming traffic (including malicious traffic), resolvers still require appropriate internal resource management mechanisms (e.g., rate limiting requests from a single source) to prevent self-overload.

* Logging Security: Logging is vital for monitoring and troubleshooting but can also leak information or be used in attacks. Control over log content, verbosity, and rate is necessary.

* Randomness Requirements: Strong source port and TXID randomization remain crucial for defending against DNS cache poisoning (see [DAGON-0X20]) and are complementary to the specifications herein.

* DNSSEC: DNS Security Extensions (DNSSEC) [RFC4033] provide cryptographic validation of DNS data origin and integrity. The pre-processing steps defined in this document should occur before or in parallel with DNSSEC validation, as even packets claiming to be DNSSEC-signed can be malformed at lower layers.

* Evolving Threats: Attack techniques evolve. DNS implementers and operators need to remain vigilant and stay informed about new security threats and mitigation strategies.

7. Relationship to Other RFCs

This document is intended to clarify and supplement existing DNS specifications, not to replace them.

* It builds upon the DNS protocol foundations defined in [RFC1034] and [RFC1035], providing more specific guidance on the response handling aspects.

* It aligns with the robustness requirements for host application layers in [RFC1122] and [RFC1123].

* It supports the measures for making DNS more resilient against forged answers proposed in [RFC5452].

* For QDCOUNT handling, it refers to the spirit of [draft-ietf-dnsop-qdcount-is-one].

8. IANA Considerations

This document does not require any IANA actions.

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[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>.

[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>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[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

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[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>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[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>.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <https://www.rfc-editor.org/info/rfc5927>.

[TUDOOR] Li, X., Xu, W., Liu, B., Zhang, M., Li, Z., Zhang, J., Chang, D., Zheng, X., Wang, C., Chen, J., Duan, H., and Q. Li, "TUDOOR Attack: Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets", Proceedings of the IEEE Symposium on Security and Privacy (S&P), DOI 10.1109/SP54039.2024.00109, May 2024, <https://ieeexplore.ieee.org/document/10540884>.

[draft-ietf-dnsop-qdcount-is-one] Hoffman, P., "Clarifying that QDCOUNT is One in DNS Responses", Work in Progress, Internet-Draft, draft-ietf-dnsop-qdcount-is-one-01, March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-qdcount-is-one-01>.

[DAGON-0X20] Dagon, D., Antonakakis, M., Vixie, P., Jinmei, T., and W. Lee, "Increased DNS Forgery Resistance through 0x20-bit

Encoding: Security via Leet Queries", Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08), DOI 10.1145/1455770.1455774, October 2008, <https://doi.org/10.1145/1455770.1455774>.

[KAMINSKY08]
Kaminsky, D., "It's The End Of The Cache As We Know It", Black Hat USA Briefings, August 2008.

[SAD-DNS] Man, K., Qian, Z., Wang, Z., Zheng, X., Huang, Y., and H. Duan, "DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels", Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), DOI 10.1145/3372297.3417280, November 2020, <https://doi.org/10.1145/3372297.3417280>.

[FRAG-POISON]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", 2013 IEEE Conference on Communications and Network Security (CNS), DOI 10.1109/CNS.2013.6682691, October 2013, <https://doi.org/10.1109/CNS.2013.6682691>.

Authors' Addresses

Xiang Li
Nankai University
38 Tongyan Road
Tianjin
Tianjin, 300355
China
Email: lixiang@nankai.edu.cn

Yuqi Qiu
Nankai University
38 Tongyan Road
Tianjin
Tianjin, 300355
China
Email: norahqiu@163.com