Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: October 7, 2020

M. Andrews
R. Bellis
ISC

April 5, 2020

A Common Operational Problem in DNS Servers - Failure To Communicate
draft-ietf-dnsop-no-response-issue-19

Abstract

The DNS is a query / response protocol. Failing to respond to queries, or responding incorrectly, causes both immediate operational problems and long term problems with protocol development.

This document identifies a number of common kinds of queries to which some servers either fail to respond or else respond incorrectly. This document also suggests procedures for zone operators to apply to identify and remediate the problem.

The document does not look at the DNS data itself, just the structure of the responses.

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 October 7, 2020.

Table of Contents

1. Introduction

The DNS [RFC1034], [RFC1035] is a query / response protocol. Failing to respond to queries, or responding incorrectly, causes both immediate operational problems and long term problems with protocol development.

Failure to respond to a query is indistinguishable from packet loss without doing an analysis of query-response patterns. Additionally failure to respond results in unnecessary queries being made by DNS clients, and introduces delays to the resolution process.

Due to the inability to distinguish between packet loss and nameservers dropping EDNS [RFC6891] queries, packet loss is sometimes misclassified as lack of EDNS support which can lead to DNSSEC validation failures.

The existence of servers which fail to respond to queries results in developers being hesitant to deploy new standards. Such servers need to be identified and remediated.

The DNS has response codes that cover almost any conceivable query response. A nameserver should be able to respond to any conceivable query using them. There should be no need to drop queries because a nameserver does not understand them.

Unless a nameserver is under attack, it should respond to all DNS requests directed to it. When a nameserver is under attack it may wish to drop packets. A common attack is to use a nameserver as an amplifier by sending spoofed packets. This is done because response packets are bigger than the queries and large amplification factors are available especially if EDNS is supported. Limiting the rate of responses is reasonable when this is occurring and the client should retry. This however only works if legitimate clients are not being forced to guess whether EDNS queries are accepted or not. While there is still a pool of servers that don't respond to EDNS requests,

clients have no way to know if the lack of response is due to packet loss, or EDNS packets not being supported, or rate limiting due to the server being under attack. Misclassification of server behaviour is unavoidable when rate limiting is used until the population of servers which fail to respond to well-formed queries drops to near zero.

Nameservers should respond to queries even if the queried name is not for any name the server is configured to answer for. Misconfigured nameservers are a common occurrence in the DNS and receiving queries for zones that the server is not configured for is not necessarily an indication that the server is under attack. Parent zone operators are advised to regularly check that the delegating NS records are consistent with those of the delegated zone and to correct them when they are not [RFC1034]. Doing this regularly should reduce the instances of broken delegations.

This document does not try to identify all possible errors nor does it supply an exhaustive list of tests.

2. Consequences

Failure to follow the relevant DNS RFCs has multiple adverse consequences. Some are caused directly from the non-compliant behaviour and others as a result of work-arounds forced on recursive servers. Addressing known issues now will reduce future interoperability issues as the DNS protocol continues to evolve and clients make use of newly-introduced DNS features. In particular the base DNS specification [RFC1034], [RFC1035] and the EDNS specification [RFC6891], when implemented, need to be followed.

Some examples of known consequences include:

- The AD flag bit in a response cannot be trusted to mean anything as some servers incorrectly copy the flag bit from the request to the response [RFC1035], [RFC4035].

- Widespread non-response to EDNS queries has lead to recursive servers having to assume that EDNS is not supported and that fallback to plain DNS is required, potentially causing DNSSEC validation failures.

- Widespread non-response to EDNS options, requires recursive servers to have to decide whether to probe to see if it is the EDNS option or just EDNS that is causing the non response. In the limited amount of time required to resolve a query before the client times out this is not possible.

- Incorrectly returning FORMERR to an EDNS option being present, leads to the recursive server not being able to determine if the server is just broken in the handling of the EDNS option or doesn't support EDNS at all.

- Mishandling of unknown query types has contributed to the abandonment of the transition of the SPF type.

- Mishandling of unknown query types has slowed up the development of DANE and resulted in additional rules being specified to reduce the probability of interacting with a broken server when making TLSA queries.

The consequences of servers not following the RFCs will only grow if measures are not put in place to remove non compliant servers from the ecosystem. Working around issues due to non-compliance with RFCs is not sustainable.

Most (if not all) of these consequences could have been avoided if action had been taken to remove non-compliant servers as soon as people were aware of them, i.e. to actively seek out broken implementations and servers and inform their developers and operators that they need to fix their servers.

3. Common kinds of queries that result in no or bad responses.

This section is broken down into Basic DNS requests and EDNS requests.

3.1. Basic DNS Queries

3.1.1. Zone Existence

If a zone is delegated to a server, that server should respond to an SOA query for that zone with an SOA record. Failing to respond at all is always incorrect, regardless of the configuration of the server. Responding with anything other than an SOA record in the Answer section indicates a bad delegation.

3.1.2. Unknown / Unsupported Type Queries

Some servers fail to respond to unknown or unsupported types. If a server receives a query for a type that it doesn't recognise, or doesn't implement, it is expected to return the appropriate response as if it did recognise the type but does not have any data for that type: either NOERROR, or NXDOMAIN. The exception to this are queries for Meta-RR types which may return NOTIMP.

3.1.3. DNS Flags

Some servers fail to respond to DNS queries with various DNS flags set, regardless of whether they are defined or still reserved. At the time of writing there are servers that fail to respond to queries with the AD bit set to 1 and servers that fail to respond to queries with the last reserved flag bit set.

Servers should respond to such queries. If the server does not know the meaning of a flag bit it must not copy it to the response [RFC1035] Section 4.1.1. If the server does not understand the meaning of a request it should reply with a FORMERR response with unknown flags set to zero.

3.1.3.1. Recursive Queries

A non-recursive server is supposed to respond to recursive queries as if the RD bit is not set [RFC1034].

3.1.4. Unknown DNS opcodes

The use of previously undefined opcodes is to be expected. Since the DNS was first defined two new opcodes have been added, UPDATE and NOTIFY.

NOTIMP is the expected rcode to an unknown or unimplemented opcode.

Note: while new opcodes will most probably use the current layout structure for the rest of the message there is no requirement that anything other than the DNS header match.

3.1.5. TCP Queries

All DNS servers are supposed to respond to queries over TCP [RFC7766]. While firewalls should not block TCP connection attempts if they do they should cleanly terminate the connection by sending TCP RESET or sending ICMP/ICMPv6 Administratively Prohibited messages. Dropping TCP connections introduces excessive delays to the resolution process.

3.2. EDNS Queries

EDNS queries are specified in [RFC6891].

3.2.1. EDNS Queries - Version Independent

Identifying servers that fail to respond to EDNS queries can be done by first confirming that the server responds to regular DNS queries, followed by a series of otherwise identical queries using EDNS, then making the original query again. A series of EDNS queries is needed as at least one DNS implementation responds to the first EDNS query with FORMERR but fails to respond to subsequent queries from the same address for a period until a regular DNS query is made. The EDNS query should specify a UDP buffer size of 512 bytes to avoid false classification of not supporting EDNS due to response packet size.

If the server responds to the first and last queries but fails to respond to most or all of the EDNS queries, it is probably faulty. The test should be repeated a number of times to eliminate the likelihood of a false positive due to packet loss.

Firewalls may also block larger EDNS responses but there is no easy way to check authoritative servers to see if the firewall is mis-configured.

3.2.2. EDNS Queries - Version Specific

Some servers respond correctly to EDNS version 0 queries but fail to respond to EDNS queries with version numbers that are higher than zero. Servers should respond with BADVERS to EDNS queries with version numbers that they do not support.

Some servers respond correctly to EDNS version 0 queries but fail to set QR=1 when responding to EDNS versions they do not support. Such responses may be discarded as invalid (as QR is not 1) or treated as requests (when the source port of the original request was port 53).

3.2.3. EDNS Options

Some servers fail to respond to EDNS queries with EDNS options set. The original EDNS specification left this behaviour undefined [RFC2671], but the correct behaviour was clarified in [RFC6891]. Unknown EDNS options are supposed to be ignored by the server.

3.2.4. EDNS Flags

Some servers fail to respond to EDNS queries with EDNS flags set. Servers should ignore EDNS flags they do not understand and must not add them to the response [RFC6891].

3.2.5. Truncated EDNS Responses

Some EDNS aware servers fail to include an OPT record when a truncated response is sent. An OPT record is supposed to be included in a truncated response [RFC6891].

Some EDNS aware servers fail to honour the advertised EDNS UDP buffer size and send over-sized responses [RFC6891]. Servers must send UDP responses no larger than the advertised EDNS UDP buffer size.

3.2.6. DO=1 Handling

Some nameservers incorrectly only return an EDNS response when the DO bit [RFC3225] is 1 in the query. Servers that support EDNS should always respond to EDNS requests with EDNS responses.

Some nameservers fail to copy the DO bit to the response despite clearly supporting DNSSEC by returning an RRSIG records to EDNS queries with DO=1.

3.2.7. EDNS over TCP

Some EDNS aware servers incorrectly limit the TCP response sizes to the advertised UDP response size. This breaks DNS resolution to clients where the response sizes exceed the advertised UDP response size despite the server and the client being capable of sending and receiving larger TCP responses respectively. It effectively defeats setting TC=1 in UDP responses.

4. Firewalls and Load Balancers

Firewalls and load balancers can affect the externally visible behaviour of a nameserver. Tests for conformance should to be done from outside of any firewall so that the system is tested as a whole.

Firewalls and load balancers should not drop DNS packets that they don't understand. They should either pass the packets or generate an appropriate error response.

Requests for unknown query types are normal client behaviour and should not be construed as an attack. Nameservers have always been expected to be able to handle such queries.

Requests for unknown query classes are normal client behaviour and should not be construed as an attack. Nameservers have always been expected to be able to handle such queries.

Requests with unknown opcodes are normal client behaviour and should not be construed as an attack. Nameservers have always been expected to be able to handle such queries.

Requests with unassigned flags set (DNS or EDNS) are expected client behaviour and should not be construed as an attack. The behaviour for unassigned flags is to ignore them in the request and to not set them in the response. Dropping DNS / EDNS packets with unassigned flags makes it difficult to deploy extensions that make use of them due to the need to reconfigure and update firewalls.

Requests with unknown EDNS options are expected client behaviour and should not be construed as an attack. The correct behaviour for unknown EDNS options is to ignore their presence when constructing a reply.

Requests with unknown EDNS versions are expected client behaviour and should not be construed as an attack. The correct behaviour for unknown EDNS versions is to return BADVERS along with the highest EDNS version the server supports. Dropping EDNS packets breaks EDNS version negotiation.

Firewalls should not assume that there will only be a single response message to a request. There have been proposals to use EDNS to signal that multiple DNS messages be returned rather than a single UDP message that is fragmented at the IP layer.

DNS, and EDNS in particular, are designed to allow clients to be able to use new features against older servers without having to validate every option. Indiscriminate blocking of messages breaks that design.

However, there may be times when a nameserver mishandles messages with a particular flag, EDNS option, EDNS version field, opcode, type or class field or combination thereof to the point where the integrity of the nameserver is compromised. Firewalls should offer the ability to selectively reject messages using an appropriately constructed response based on all these fields while awaiting a fix from the nameserver vendor.

5. Packet Scrubbing Services

Packet scrubbing services are used to filter out undesired traffic, including but not limited to, denial of service traffic. This is often done using heuristic analysis of the traffic.

Packet scrubbing services can affect the externally visible behaviour of a nameserver in a similar way to firewalls. If an operator uses a

packet scrubbing service, they should check that legitimate queries are not being blocked.

Packet scrubbing services, unlike firewalls, are also turned on and off in response to denial of service attacks. One needs to take care when choosing a scrubbing service.

Ideally, Operators should run these tests against a packet scrubbing service to ensure that these tests are not seen as attack vectors.

6. Whole Answer Caches

Whole answer caches take a previously constructed answer and return it to a subsequent query for the same question. However, they can return the wrong response if they do not take all of the relevant attributes of the query into account.

In addition to the standard tuple of <qname,qtype,qclass> a non-exhaustive set of attributes that must be considered include: RD, AD, CD, OPT record, DO, EDNS buffer size, EDNS version, EDNS options, and transport.

7. Response Code Selection

Choosing the correct response code when responding to DNS queries is important. Response codes should be chosen considering how clients will handle them.

For unimplemented opcodes NOTIMP is the expected response code. Note: Newly implemented opcodes may change the message format by extending the header, changing the structure of the records, etc. Servers are not expected to be able to parse these, and should respond with a response code of NOTIMP rather than FORMERR (which would be expected if there was a parse error with an known opcode).

For unimplemented type codes, and in the absence of other errors, the only valid response is NoError if the qname exists, and NameError (NXDOMAIN) otherwise. For Meta-RRs NOTIMP may be returned instead.

If a zone cannot be loaded because it contains unimplemented type codes that are not encoded as unknown record types according to [RFC3597] then the expected response is SERVFAIL as the whole zone should be rejected Section 5.2 [RFC1035]. If a zone loads then Section 4.3.2 [RFC1034] applies.

If the server supports EDNS and receives a query with an unsupported EDNS version, the correct response is BADVERS [RFC6891].

If the server does not support EDNS at all, FORMERR is the expected error code. That said a minimal EDNS server implementation requires parsing the OPT records and responding with an empty OPT record in the additional section in most cases. There is no need to interpret any EDNS options present in the request as unsupported EDNS options are expected to be ignored [RFC6891]. Additionally EDNS flags can be ignored. The only part of the OPT record that needs to be examined is the version field to determine if BADVERS needs to be sent or not.

8. Testing

Testing is divided into two sections. "Basic DNS", which all servers should meet, and "Extended DNS", which should be met by all servers that support EDNS (a server is deemed to support EDNS if it gives a valid EDNS response to any EDNS query). If a server does not support EDNS it should still respond to all the tests.

These tests query for records at the apex of a zone that the server is nominally configured to serve. All tests should use the same zone.

It is advisable to run all of the tests below in parallel so as to minimise the delays due to multiple timeouts when the servers do not respond. There are 16 queries directed to each nameserver (assuming no packet loss) testing different aspects of Basic DNS and Extended DNS.

The tests below use dig from BIND 9.11.0.

When testing recursive servers set RD=1 and choose a zone name that is know to exist and is not being served by the recursive server. The root zone (".") is often a good candidate as it is DNSSEC signed. RD=1, rather than RD=0, should be present in the responses for all test involving the opcode QUERY. Non-authoritative answers (AA=0) are expected when talking to a recursive server. AD=1 is only expected if the server is validating responses and one or both AD=1 or DO=1 is set in the request otherwise AD=0 is expected.

8.1. Testing - Basic DNS

This first set of tests cover basic DNS server behaviour and all servers should pass these tests.

8.1.1. Is The Server Configured For The Zone?

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set and without EDNS.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We do not expect an OPT record to be returned [RFC6891].

Verify the server is configured for the zone:

```
dig +noedns +noad +norec soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present
```

8.1.2. Testing Unknown Types

Identifying servers that fail to respond to unknown or unsupported types can be done by making an initial DNS query for an A record, making a number of queries for an unallocated type, then making a query for an A record again. IANA maintains a registry of allocated types.

If the server responds to the first and last queries but fails to respond to the queries for the unallocated type, it is probably faulty. The test should be repeated a number of times to eliminate the likelihood of a false positive due to packet loss.

Ask for the TYPE1000 RRset at the configured zone's name. This query is made with no DNS flag bits set and without EDNS. TYPE1000 has been chosen for this purpose as IANA is unlikely to allocate this type in the near future and it is not in a range reserved for private use [RFC6895]. Any unallocated type code could be chosen for this test.

We expect no records to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We do not expect an OPT record to be returned [RFC6891].

Check that queries for an unknown type work:

dig +noedns +noad +norec type1000 $zone @$server

expect: status: NOERROR
expect: an empty answer section.
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present

8.1.3. Testing Header Bits

8.1.3.1. Testing CD=1 Queries

Ask for the SOA record of the configured zone. This query is made with only the CD DNS flag bit set, all other DNS bits clear, and without EDNS.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header. We do not expect an OPT record to be returned.

If the server supports DNSSEC, CD should be set in the response [RFC4035] otherwise CD should be clear [RFC1034].

Check that queries with CD=1 work:

dig +noedns +noad +norec +cd soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present

8.1.3.2. Testing AD=1 Queries

Ask for the SOA record of the configured zone. This query is made with only the AD DNS flag bit set and all other DNS bits clear and without EDNS.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header. We do not expect an OPT record to be returned. The purpose of this query is to detect blocking of queries with the AD bit present, not the specific value of AD in the response.

Check that queries with AD=1 work:

dig +noedns +norec +ad soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: the OPT record to NOT be present

AD use in queries is defined in [RFC6840].

8.1.3.3. Testing Reserved Bit

Ask for the SOA record of the configured zone. This query is made with only the final reserved DNS flag bit set and all other DNS bits clear and without EDNS.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may be set. The final reserved bit must not be set [RFC1034]. We do not expect an OPT record to be returned [RFC6891].

Check that queries with the last unassigned DNS header flag work and that the flag bit is not copied to the response:

dig +noedns +noad +norec +zflag soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: MBZ to NOT be in the response (see below)
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present

MBZ (Must Be Zero) is a dig-specific indication that the flag bit has been incorrectly copied. See Section 4.1.1, [RFC1035] "Z Reserved for future use. Must be zero in all queries and responses."

8.1.3.4. Testing Recursive Queries

Ask for the SOA record of the configured zone. This query is made with only the RD DNS flag bit set and without EDNS.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA, QR and RD bits

to be set in the header; RA may also be set [RFC1034]. We do not expect an OPT record to be returned [RFC6891].

Check that recursive queries work:

dig +noedns +noad +rec soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: flag: aa to be present
expect: flag: rd to be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present

8.1.4. Testing Unknown Opcodes

Construct a DNS message that consists of only a DNS header with opcode set to 15 (currently not allocated), no DNS header bits set and empty question, answer, authority and additional sections.

Check that new opcodes are handled:

dig +noedns +noad +opcode=15 +norec +header-only @$server

expect: status: NOTIMP
expect: opcode: 15
expect: all sections to be empty
expect: flag: aa to NOT be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present

8.1.5. Testing TCP

Whether a server accepts TCP connections can be tested by first checking that it responds to UDP queries to confirm that it is up and operating, then attempting the same query over TCP. An additional query should be made over UDP if the TCP connection attempt fails to confirm that the server under test is still operating.

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set and without EDNS. This query is to be sent using TCP.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We do not expect an OPT record to be returned [RFC6891].

Check that TCP queries work:

```
dig +noedns +noad +norec +tcp soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: flag: aa to be present
expect: flag: rd to NOT be present
expect: flag: ad to NOT be present
expect: the OPT record to NOT be present
```

The requirement that TCP be supported is defined in [RFC7766].

8.2. Testing - Extended DNS

The next set of tests cover various aspects of EDNS behaviour. If any of these tests succeed (indicating at least some EDNS support) then all of them should succeed. There are servers that support EDNS but fail to handle plain EDNS queries correctly so a plain EDNS query is not a good indicator of lack of EDNS support.

8.2.1. Testing Minimal EDNS

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 0 is used without any EDNS options or EDNS flags set.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response. The EDNS version field should be 0 and there should be no EDNS options present [RFC6891].

Check that plain EDNS queries work:

```
dig +nocookie +edns=0 +noad +norec soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: an OPT record to be present in the additional section
expect: EDNS Version 0 in response
expect: flag: aa to be present
expect: flag: ad to NOT be present
```

+nocookie disables sending a EDNS COOKIE option which is otherwise enabled by default in BIND 9.11.0 (and later).

8.2.2. Testing EDNS Version Negotiation

Ask for the SOA record of a zone the server is nominally configured to serve. This query is made with no DNS flag bits set. EDNS version 1 is used without any EDNS options or EDNS flags set.

We expect the SOA record for the zone to NOT be returned in the answer section with the extended rcode set to BADVERS and the QR bit to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response. The EDNS version field should be 0 in the response as no other EDNS version has as yet been specified [RFC6891].

Check that EDNS version 1 queries work (EDNS supported):

```
dig +nocookie +edns=1 +noednsneg +noad +norec soa $zone @$server
```

```
expect: status: BADVERS
expect: the SOA record to NOT be present in the answer section
expect: an OPT record to be present in the additional section
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
expect: flag: ad to NOT be present
```

+noednsneg has been set as dig supports EDNS version negotiation and we want to see only the response to the initial EDNS version 1 query.

8.2.3. Testing Unknown EDNS Options

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 0 is used without any EDNS flags. An EDNS option is present with a value that has not yet been assigned by IANA. We have picked an unassigned code of 100 for the example below. Any unassigned EDNS option code could have been choosen for this test.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response. The EDNS version field should be 0 as EDNS versions other than 0 are yet to be specified and there should be no EDNS options present as unknown EDNS options are supposed to be ignored by the server [RFC6891] Section 6.1.2.

Check that EDNS queries with an unknown option work (EDNS supported):

```
dig +nocookie +edns=0 +noad +norec +ednsopt=100 soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: an OPT record to be present in the additional section
expect: OPT=100 to NOT be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
expect: flag: ad to NOT be present
```

8.2.4. Testing Unknown EDNS Flags

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 0 is used without any EDNS options. An unassigned EDNS flag bit is set (0x40 in this case).

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response as unknown EDNS flags are supposed to be ignored. The EDNS version field should be 0 and there should be no EDNS options present [RFC6891].

Check that EDNS queries with unknown flags work (EDNS supported):

```
dig +nocookie +edns=0 +noad +norec +ednsflags=0x40 soa $zone @$server

expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: an OPT record to be present in the additional section
expect: MBZ not to be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
expect: flag: ad to NOT be present
```

MBZ (Must Be Zero) is a dig-specific indication that a flag bit has been incorrectly copied as per Section 6.1.4, [RFC6891].

8.2.5. Testing EDNS Version Negotiation With Unknown EDNS Flags

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 1 is used without any EDNS options. An unassigned EDNS flag bit is set (0x40 in this case).

We expect the SOA record for the zone to NOT be returned in the answer section with the extended rcode set to BADVERS and the QR bit to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response as unknown EDNS flags are supposed to be ignored. The EDNS version field should be 0 as EDNS versions other than 0 are yet to be specified and there should be no EDNS options present [RFC6891].

Check that EDNS version 1 queries with unknown flags work (EDNS supported):

```
dig +nocookie +edns=1 +noednsneg +noad +norec +ednsflags=0x40 soa \
    $zone @$server

expect: status: BADVERS
expect: SOA record to NOT be present
expect: an OPT record to be present in the additional section
expect: MBZ not to be present
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
expect: flag: ad to NOT be present
```

8.2.6. Testing EDNS Version Negotiation With Unknown EDNS Options

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 1 is used. An unknown EDNS option is present. We have picked an unassigned code of 100 for the example below. Any unassigned EDNS option code could have been chosen for this test.

We expect the SOA record for the zone to NOT be returned in the answer section with the extended rcode set to BADVERS and the QR bit to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response. The EDNS version field should be 0 as EDNS versions other than 0 are yet to be specified and there should be no EDNS options present [RFC6891].

Check that EDNS version 1 queries with unknown options work (EDNS supported):

```
dig +nocookie +edns=1 +noednsneg +noad +norec +ednsopt=100 soa \
    $zone @$server

expect: status: BADVERS
expect: SOA record to NOT be present
expect: an OPT record to be present in the additional section
expect: OPT=100 to NOT be present
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
expect: flag: ad to NOT be present
```

8.2.7. Testing Truncated Responses

Ask for the DNSKEY records of the configured zone, which must be a DNSSEC signed zone. This query is made with no DNS flag bits set. EDNS version 0 is used without any EDNS options. The only EDNS flag set is DO. The EDNS UDP buffer size is set to 512. The intention of this query is to elicit a truncated response from the server. Most signed DNSKEY responses are bigger than 512 bytes. This test will not give a valid result if the zone is not signed.

We expect a response, the rcode to be set to NOERROR, and the AA and QR bits to be set, AD may be set in the response if the server supports DNSSEC otherwise it should be clear; TC and RA may also be set [RFC1035] [RFC4035]. We expect an OPT record to be present in the response. There should be no EDNS flags other than DO present in the response. The EDNS version field should be 0 and there should be no EDNS options present [RFC6891].

If TC is not set it is not possible to confirm that the server correctly adds the OPT record to the truncated responses or not.

```
dig +norec +dnssec +bufsize=512 +ignore dnskey $zone @$server
expect: NOERROR
expect: OPT record with version set to 0
```

8.2.8. Testing DO=1 Handling

Ask for the SOA record of the configured zone, which does not need to be DNSSEC signed. This query is made with no DNS flag bits set. EDNS version 0 is used without any EDNS options. The only EDNS flag set is DO.

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be

set in the response, AD may be set in the response if the server supports DNSSEC otherwise it should be clear; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags other than DO present in the response which should be present if the server supports DNSSEC. The EDNS version field should be 0 and there should be no EDNS options present [RFC6891].

Check that DO=1 queries work (EDNS supported):

```
dig +nocookie +edns=0 +noad +norec +dnssec soa $zone @$server
```

```
expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: an OPT record to be present in the additional section
expect: DO=1 to be present if an RRSIG is in the response
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

8.2.9. Testing EDNS Version Negotiation With DO=1

Ask for the SOA record of the configured zone, which does not need to be DNSSEC signed. This query is made with no DNS flag bits set. EDNS version 1 is used without any EDNS options. The only EDNS flag set is DO.

We expect the SOA record for the zone to NOT be returned in the answer section, the rcode to be set to NOERROR, ; the QR bit and possibly the RA bit to be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags other than DO present in the response which should be there if the server supports DNSSEC. The EDNS version field should be 0 and there should be no EDNS options present [RFC6891].

Check that EDNS version 1, DO=1 queries work (EDNS supported):

```
dig +nocookie +edns=1 +noednsneg +noad +norec +dnssec soa \
    $zone @$server
```

```
expect: status: BADVERS
expect: SOA record to NOT be present
expect: an OPT record to be present in the additional section
expect: DO=1 to be present if the EDNS version 0 DNSSEC query test
        returned DO=1
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
```

8.2.10. Testing With Multiple Defined EDNS Options

Ask for the SOA record of the configured zone. This query is made with no DNS flag bits set. EDNS version 0 is used. A number of defined EDNS options are present (NSID [RFC5001], DNS COOKIE [RFC7873], EDNS Client Subnet [RFC7871] and EDNS Expire [RFC7314]).

We expect the SOA record for the zone to be returned in the answer section, the rcode to be set to NOERROR, and the AA and QR bits to be set in the header; RA may also be set [RFC1034]. We expect an OPT record to be returned. There should be no EDNS flags present in the response. The EDNS version field should be 0. Any of the requested EDNS options supported by the server and permitted server configuration may be returned [RFC6891].

Check that EDNS queries with multiple defined EDNS options work:

```
dig +edns=0 +noad +norec +cookie +nsid +expire +subnet=0.0.0.0/0 \
    soa $zone @$server
```

```
expect: status: NOERROR
expect: the SOA record to be present in the answer section
expect: an OPT record to be present in the additional section
expect: EDNS Version 0 in response
expect: flag: aa to be present
expect: flag: ad to NOT be present
```

8.3. When EDNS Is Not Supported

If EDNS is not supported by the nameserver, we expect a response to each of the above queries. That response may be a FORMERR error response or the OPT record may just be ignored.

Some nameservers only return a EDNS response when a particular EDNS option or flag (e.g. DO=1) is present in the request. This behaviour is not compliant behaviour and may hide other incorrect behaviour from the above tests. Re-testing with the triggering option / flag present will expose this misbehaviour.

9. Remediation

Name server operators are generally expected to test their own infrastructure for compliance to standards. The above tests should be run when new systems are brought online, and should be repeated periodically to ensure continued interoperability.

Domain registrants who do not maintain their own DNS infrastructure are entitled to a DNS service that conforms to standards and

interoperates well. Registrants who become aware that their DNS operator does not have a well maintained or compliant infrastructure should insist that their service provider correct issues, and switch providers if they do not.

In the event that an operator experiences problems due to the behaviour of name servers outside their control, the above tests will help in narrowing down the precise issue(s) which can then be reported to the relevant party.

If contact information for the operator of a misbehaving name server is not already known, the following methods of communication could be considered:

- the RNAME of the zone authoritative for the name of the misbehaving server
- the RNAME of zones for which the offending server is authoritative
- administrative or technical contacts listed in the registration information for the parent domain of the name of the misbehaving server, or for zones for which the name server is authoritative
- the registrar or registry for such zones
- DNS-specific operational fora (e.g. mailing lists)

Operators of parent zones may wish to regularly test the authoritative name servers of their child zones. However, parent operators can have widely varying capabilities in terms of notification or remediation depending on whether they have a direct relationship with the child operator. Many TLD registries, for example, cannot directly contact their registrants and may instead need to communicate through the relevant registrar. In such cases it may be most efficient for registrars to take on the responsibility for testing the name servers of their registrants, since they have a direct relationship.

When notification is not effective at correcting problems with a misbehaving name server, parent operators can choose to remove NS record sets (and glue records below) that refer to the faulty server until the servers are fixed. This should only be done as a last resort and with due consideration, as removal of a delegation can have unanticipated side effects. For example, other parts of the DNS tree may depend on names below the removed zone cut, and the parent operator may find themselves responsible for causing new DNS failures to occur.

10. Security Considerations

Testing protocol compliance can potentially result in false reports of attempts to break services from Intrusion Detection Services and firewalls. All of the tests are well-formed (though not necessarily common) DNS queries. None of the tests listed above should cause any harm to a protocol-compliant server.

Relaxing firewall settings to ensure EDNS compliance could potentially expose a critical implementation flaw in the nameserver. Nameservers should be tested for conformance before relaxing firewall settings.

When removing delegations for non-compliant servers there can be a knock on effect on other zones that require these zones to be operational for the nameservers addresses to be resolved.

11. IANA Considerations

There are no actions for IANA.

12. Acknowledgements

The contributions of the following are gratefully acknowledged:

Matthew Pounsett, Tim Wicinski.

13. References

13.1. Normative References

[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>.

[RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, DOI 10.17487/RFC3225, December 2001, <https://www.rfc-editor.org/info/rfc3225>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <https://www.rfc-editor.org/info/rfc6840>.

[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>.

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <https://www.rfc-editor.org/info/rfc6895>.

[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>.

13.2. Informative References

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, DOI 10.17487/RFC2671, August 1999, <https://www.rfc-editor.org/info/rfc2671>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC5001] Austein, R., "DNS Name Server Identifier (NSID) Option", RFC 5001, DOI 10.17487/RFC5001, August 2007, <https://www.rfc-editor.org/info/rfc5001>.

[RFC7314] Andrews, M., "Extension Mechanisms for DNS (EDNS) EXPIRE Option", RFC 7314, DOI 10.17487/RFC7314, July 2014, <https://www.rfc-editor.org/info/rfc7314>.

[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>.

[RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <https://www.rfc-editor.org/info/rfc7873>.

Authors' Addresses

M. Andrews
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: marka@isc.org

Ray Bellis
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: ray@isc.org