Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 22, 2017

M. Andrews
ISC

September 18, 2016

A Common Operational Problem in DNS Servers - Failure To Respond.
draft-ietf-dnsop-no-response-issue-05

Abstract

The DNS is a query / response protocol. Failure to respond or to respond correctly to queries causes both immediate operational problems and long term problems with protocol development.

This document identifies a number of common kinds of queries which some servers either fail to respond or else respond incorrectly. This document also suggests procedures for TLD and other zone operators to apply to help reduce / eliminate 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 http://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 March 22, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

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

Failure to respond to a query is indistinguishable from a packet loss. Without doing a analysis of query response patterns will results in unnecessary additional queries being made by DNS clients, and delays being introduced 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.

Servers which fail to respond to queries to remain results in developers being hesitant to deploy new standards. Such servers need to be identified.

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.

Unless a nameserver is under attack, it should respond to all queries directed to it. Additionally, the nameserver should not assume that there isn't a delegation to the server even if it is not configured to serve the zone. Broken 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 supposed 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.

When a nameserver is under attack it may wish to drop packets. A common attack is to use a nameserver as a amplifier by sending spoofed packets. This is done because response packets are bigger than the queries and big 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, EDNS packets not being supported or rate limiting due to the server being under attack. Mis-classifications of server characteristics are unavoidable when rate limiting is done.

2. Consequences

Lack of following the relevant RFCs has lead to various consequences. Some as a direct result and some from recursive servers try to work around the non compliance. Fixing known issues know will reduce future consequences as DNS clients make use of the features available in the DNS protocol.

The AD flag bit in a response cannot be trusted to mean anything as servers incorrectly copied the flag bit from the request to the response despite the prohibition.

Wide spread non response to EDNS queries has lead to recursive servers having to assume EDNS may not supported and fallback to plain DNS is required. Servers get incorrectly diagnosed as not supporting EDNS and when they also serve signed zones DNSSEC validation fails.

Similarly, wide spread 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.

Similarly, incorrectly returning FORMERR to a 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.

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

3. Common queries kinds that result in non responses.

There are a number common query kinds that fail to respond today. They are: EDNS queries with and without extensions; queries for

unknown (unallocated) or unsupported types; and filtering of TCP queries.

3.1. Basic DNS Queries

3.1.1. Zone Existence

Initially to test existence of the zone, an SOA query should be made. If the SOA record is not returned but some other response is returned, this is a indication of a bad delegation. If the server fails to get a response to a SOA query, the Operator should make an A query for the zone, as some nameservers fail to respond to SOA queries but will respond to A queries.

3.1.2. Unknown / Unsupported Type Queries

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.

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.

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]. Firewalls that drop TCP connection attempts, they should reset the connect attempt or send a ICMP/ICMPv6 administratively prohibited message. Dropping TCP connections introduces excessive delays to the resolution process.

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.

3.2. EDNS Queries

3.2.1. EDNS Queries - Version Independent

Identifying servers that fail to respond to EDNS queries can be done by first identifying 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 answers are discarded or treated as requests.

3.2.3. EDNS Options

Some servers fail to respond to EDNS queries with EDNS options set. Unknown EDNS options are supposed to be ignored by the server [RFC6891].

3.2.4. EDNS Flags

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

3.2.5. DNSSEC

Servers should be checked to see if they support DNSSEC. Servers should also be checked to see if they support DNSSEC with EDNS.

4. Remediating

While the first step in remediating this problem is to get the offending nameserver code corrected, there is a very long tail problem with DNS servers in that it can often take over a decade between the code being corrected and a nameserver being upgraded with corrected code. With that in mind it is requested that TLD, and other similar zone operators, take steps to identify and inform their customers, directly or indirectly through registrars, that they are running such servers and that the customers need to correct the problem.

TLD operators are being asked to do this as they, due to the nature of running a TLD and the hierarchical nature of the DNS, have access to a large numbers of nameserver names as well as contact details for the registrants of those nameservers. While it is possible to construct lists of nameservers from other sources, and that has been done to survey the state of the Internet, that doesn't give the tester the contact details necessary to inform the operators. The SOA RNAME is often invalid and whois data is obscured and / or not available which makes it infeasible for others to do this.

While this section talks about TLD operators performing this work, it may be done by registrars on behalf of the TLD operator. The intent is to ensure that the testing happens and that operators of non-compliant nameservers be informed, rather than to prescribe who does the actual testing and communication. Note: having registrars perform this testing and reporting is likely to result in duplicate reports for the same server being issued by multiple registrars.

TLD operators should construct a list of servers child zones are delegated to along with a delegated zone name. This name shall be the query name used to test the server as it is supposed to exist.

For each server the TLD operator shall make an SOA query of the delegated zone name. This should result in the SOA record being returned in the answer section. If the SOA record is not returned but some other response is returned, this is a indication of a bad delegation and the TLD operator should take whatever steps it normally takes to rectify a bad delegation. If more that one zone is delegated to the server, it should choose another zone until it finds a zone which responds correctly or it exhausts the list of zones delegated to the server.

If the server fails to get a response to a SOA query, the TLD operator should make an A query as some nameservers fail to respond to SOA queries but respond to A queries. If it gets no response to the A query, another delegated zone should be queried for as some nameservers fail to respond to zones they are not configured for. If subsequent queries find a responding zone, all delegation to this server need to be checked and rectified using the TLD's normal procedures.

Having identified a working <server, query name> tuple the TLD operator should now check that the server responds to EDNS, Unknown Query Type and TCP tests as described above. If the TLD operator finds that server fails any of the tests, the TLD operator shall take steps to inform the operator of the server that they are running a faulty nameserver and that they need to take steps to correct the matter. The TLD operator shall also record the <server, query name> for follow-up testing.

If repeated attempts to inform and get the customer to correct / replace the faulty server are unsuccessful the TLD operator shall remove all delegations to said server from the zone. Removal of delegations is the step of last resort in handling complaints as specified in [RFC1033] COMPLAINTS.

It will also be necessary for TLD operators to repeat the scans periodically. It is recommended that this be performed monthly backing off to bi-annually once the numbers of faulty servers found drops off to less than 1 in 100000 servers tested. Follow-up tests for faulty servers still need to be performed monthly.

Some operators claim that they can't perform checks at registration time. If a check is not performed at registration time, it needs to be performed within a week of registration in order to detect faulty servers swiftly.

Checking of delegations by TLD operators should be nothing new as they have been required from the very beginnings of DNS to do this [RFC1034]. Checking for compliance of nameserver operations should just be a extension of such testing.

It is recommended that TLD operators setup a test web page which performs the tests the TLD operator performs as part of their regular audits to allow nameserver operators to test that they have correctly fixed their servers. Such tests should be rate limited to avoid these pages being a denial of service vector.

Nothing in this section precludes others testing servers for protocol compliance. DNS operators should test their servers to ensure that their vendors have shipped protocol compliant products. Nameserver vendors can use these tests as a part of this release processes. Registrants can use these tests to check their DNS operators servers.

5. 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 as a whole is tested.

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 is 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 is 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 is 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) is 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 is expected client behaviour and should not be construed as an attack. The correct behaviour for

unknown EDNS options is to ignore there presence when constructing a reply.

Requests with unknown EDNS versions is 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 packet breaks EDNS version negotiation.

Firewalls should not assume that there will only be a single response message to a requests. 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.

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 there of to the point where the integrity of the nameserver is compromised. Firewalls should offer the ability to selectively reject messages with an appropriately constructed response based on all these fields while awaiting a fix from the nameserver vendor.

DNS and EDNS in particular is 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.

6. Scrubbing Services

Scrubbing services, like firewalls, can affect the externally visible behaviour of a nameserver. If a operator uses a scrubbing service, they should check that legitimate queries are not being blocked.

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 and ask questions like mentioned above.

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

7. Whole Answer Caches

Whole answer caches take a previously constructed answer and return it to a subsequent query for the same qname, qtype and qclass, just updating the query id field and possibly the qname to match the incoming query to avoid constructing each response individually.

Whole answer caches can return the wrong response to a query if they do not take all of the attributes of the query into account, rather than just some of them e.g. qname, qtype and qclass. This has implications when testing and with overall protocol compliance.

Two current examples are:

Whole answer caches that ignore the EDNS version field which results in incorrect answers to non EDNS version 0 queries being returned if they were preceded by a EDNS version 0 query for the same name and type.

Whole answer caches that ignore the EDNS options in the query resulting in options only working some of the time and/or options being returned when not requested.

8. Response Code Selection

Choosing the correct response code when responding to DNS queries is important. Just because a DNS qtype is not implemented does not mean that NOTIMP is the correct response code to return. Response codes should be chosen considering how clients will handle them.

For unimplemented opcodes NOTIMP is the expected response code. For example, a new opcode could change the message format by extending the header or changing the structure of the records etc. This may result in FORMERR being returned though NOTIMP would be more correct.

Unimplemented type codes, Name Error (NXDOMAIN) and NOERROR (no data) are the expected response codes. A server is not supposed to serve a zone which contains unsupported types ([RFC1034]) so the only thing left is return if the QNAME exists or not. NOTIMP and REFUSED are not useful responses as they force the clients to try the other authoritative servers for a zone looking for a server which will answer the query.

Meta queries may be the exception but these need to be thought about on a case by case basis.

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 and NOTIMP are the expected error codes. That said a minimal EDNS server implementation requires parsing the OPT records and responding with an empty OPT record. There is no need to interpret any EDNS options present in the request as unsupported EDNS options are expected to be ignored [RFC6891].

9. 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.

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 EDNS.

The tests below use dig from BIND 9.11.0 which is still in development.

9.1. Testing - Basic DNS

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

9.1.1. Is The Server Configured For The Zone?

Verify the server is configured for the zone:

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

expect: status: NOERROR
expect: SOA record
expect: flag: aa to be present
```

9.1.2. Testing Unknown Types?

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
```

That new types are to be expected is specified in Section 3.6, [RFC1035]. Servers that don't support a new type are expected to reject a zone that contains a unsupported type as per Section 5.2, [RFC1035]. This means that a server that does load a zone can answer questions for unknown types with NOERROR or NXDOMAIN as per Section 4.3.2, [RFC1034]. [RFC6895] later reserved distinct ranges for meta and data types which allows servers to be definitive about whether a query should be answerable from zone content or not.

9.1.3. Testing Header Bits

9.1.3.1. Testing CD=1 Queries

Check that queries with CD=1 work:

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

```
expect: status: NOERROR
expect: SOA record to be present
expect: flag: aa to be present
```

CD use in queries is defined in [RFC4035].

9.1.3.2. Testing AD=1 Queries

Check that queries with AD=1 work:

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

```
expect: status: NOERROR
expect: SOA record to be present
expect: flag: aa to be present
```

AD use in queries is defined in [RFC6840].

9.1.3.3. Testing Reserved Bit

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: SOA record to be present
expect: MBZ to not be in the response
expect: flag: aa to be present
```

MBZ (Must Be Zero) presence indicates 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."

9.1.4. Testing Unknown Opcodes

Check that new opcodes are handled:

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

expect: status: NOTIMP
expect: SOA record to not be present
expect: flag: aa to NOT be present
```

As unknown opcodes have no definition, including packet format other than there must be a DNS header present, there is only one possible rcode that make sense to return to a request with a unknown opcode and that is NOTIMP.

9.1.5. Testing TCP

Check that TCP queries work:

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

expect: status: NOERROR
expect: SOA record
expect: flag: aa to be present
```

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

9.2. Testing - Extended DNS

The next set of test cover various aspects of EDNS behaviour. If any of these tests succeed, 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.

9.2.1. Testing Minimal EDNS

Check that plain EDNS queries work:

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

expect: status: NOERROR
expect: SOA record to be present
expect: OPT record to be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

+nocookie disables sending a EDNS COOKIE option in which is on by default.

9.2.2. Testing EDNS Version Negotiation

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

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

expect: status: BADVERS
expect: SOA record to not be present
expect: OPT record to be present
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
```

Only EDNS Version 0 is currently defined so the response should always be a 0 version. This will change when EDNS version 1 is defined. BADVERS is the expected rcode if EDNS is supported as per Section 6.1.3, [RFC6891].

9.2.3. Testing Unknown EDNS Options

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: SOA record to be present
expect: OPT record to be present
expect: OPT=100 to not be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

Unknown EDNS options are supposed to be ignored, Section 6.1.2, [RFC6891].

9.2.4. Testing Unknown EDNS Flags

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: SOA record to be present
expect: OPT record to be present
expect: MBZ not to be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

MBZ (Must Be Zero) presence indicates the flag bit has been incorrectly copied as per Section 6.1.4, [RFC6891].

9.2.5. Testing EDNS Version Negotiation With Unknown EDNS Flags

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: OPT record to be present
expect: MBZ not to be present
expect: EDNS Version 0 in response
expect: flag: aa to NOT be present
```

+noednsneg disables EDNS version negotiation in DiG; MBZ (Must Be Zero) presence indicates the flag bit has been incorrectly copied.

9.2.6. Testing EDNS Version Negotiation With Unknown EDNS Options

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: OPT record to be present
expect: OPT=100 to NOT be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

+noednsneg disables EDNS version negotiation in DiG.

9.2.7. Testing DNSSEC Queries

Check that a DNSSEC queries work (EDNS supported):

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

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

DO=1 should be present if RRSIGs are returned as they indicate that the server supports DNSSEC. Servers that support DNSSEC are supposed to copy the DO bit from the request to the response as per [RFC3225].

9.2.8. Testing EDNS Version Negotiation With DNSSEC

Check that EDNS version 1 DNSSEC 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: OPT record to be present
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
```

+noednsneg disables EDNS version negotiation in DiG.

9.2.9. Testing With Multiple Defined EDNS Options

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: SOA record to be present
expect: OPT record to be present
expect: EDNS Version 0 in response
expect: flag: aa to be present
```

9.3. When EDNS Is Not Supported

If EDNS is not supported by the nameserver, we expect a response to all the above queries. That response may be a FORMERR or NOTIMP 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.

10. Security Considerations

Testing protocol compliance can potentially result in false reports of attempts to break services from Intrusion Detection Services and firewalls. None of the tests listed above should break nominally EDNS compliant servers. None of the tests above should break non EDNS servers. All the tests above are well formed, though not necessarily common, DNS queries.

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. References

12.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, DOI 10.17487/RFC3225, December 2001, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7766>.

12.2. Informative References

[RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <http://www.rfc-editor.org/info/rfc1033>.

Author's Address

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

Email: marka@isc.org