       A Common Operational Problem in DNS Servers - Failure To Respond.
                draft-andrews-dns-no-response-issue-10.txt

Abstract

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

   This document identifies a number of common classes of queries that
   some servers fail to respond too. This document also suggests
   procedures for TLD and other similar zone operators to apply to help
   reduce / eliminate the problem.

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 February 19, 2016.

Table of Contents

1. Introduction

The DNS [RFC1034], [RFC1035] is a query / response protocol. Failure
to respond to queries 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 and results in
unnecessary additional queries being made by DNS clients and
unnecessary 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.

Allowing servers which fail to respond to queries to remain results
in developers being afraid to deploy implementations of recent
standards. Such servers need to be identified and corrected /
replaced.

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 as a result of following delegations. Additionally
code should not assume that there isn't a delegation to the server
even if it is not configured to serve the zone. Broken delegation
are a common occurrence in the DNS and receiving queries for zones
that you are not configured for is not a necessarily a indication
that you are 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]. If this was being done regularly the instances of broken
delegations would be much lower.

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 accept 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](#). Common queries class that result in non responses.

There are three common query classes that result in non responses
today. These are EDNS queries, queries for unknown (unallocated) or
unsupported types and filtering of TCP queries.

[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
then making a series of otherwise identical responses 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
misconfigured.

[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
answer are discarded or treated as requests.

[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](#)].

2.4. EDNS Flags

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

2.5. 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.

2.6. 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, them 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.

2.7. 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 a unknown / unimplemented opcode.

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

2.8. TCP Queries

All DNS servers are supposed to respond to queries over TCP
[RFC5966]. Firewalls that drop TCP connection attempts rather that
resetting the connect attempt or send a ICMP/ICMPv6 administratively
prohibited message introduce 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. 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 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 the
delegated zone name. This should result in the SOA record being
returned in the answer section. If the SOA record is not return 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 a 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 followup 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.

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.

4. Firewalls and Load Balancers

   Firewalls and load balancers can affect the externally visible
   behaviour of a nameserver. Tests for conformance need 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 through the packets or
   generate a appropriate error response.

   Requests for unknown query types are not attacks and should not be
   treated as such.

   Requests with unassigned flags set (DNS or EDNS) are not attacks and
   should not be treated as such. The behaviour for unassigned is to
   ignore them in the request and to not set them in the response. All
   dropping DNS / EDNS packets with unassigned flags does is make it
   harder to deploy extension that make use of them due to the need to
   reconfigure / update firewalls.

Requests with unknown EDNS options are not a attack and should not be treated as such. The correct behaviour for unknown EDNS options is to ignore them.

Requests with unknown EDNS versions are not a attack and should not be treated as such. The correct behaviour for unknown EDNS versions is to return BADVERS along with the highest EDNS version the server supports. All dropping EDNS packets does is break 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.

5. Scrubbing Services

Scrubbing services, like firewalls, can affect the externally visible behaviour of a nameserver. If you use a scrubbing service you 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:

      do they pass unknown DNS query types.
      do they pass unknown EDNS versions.
      do they pass unknown EDNS options.
      do they pass unknown EDNS flags.
      do they pass requests with unknown DNS opcodes.
      do they pass requests with the remaining reserved DNS header flag
      bit set.

All of these are not attack vectors but some scrubbing services treat them as such.

6. Response Code Selection

Choosing the correct response code when fixing a nameserver is important. Just because a type is not implemented does not mean that NOTIMP is the correct response code to return. Response codes need to be chosen considering how clients will handle them.

For unimplemented opcodes NOTIMP is the expected response code. Additionally 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.

In general, for 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 all the authoritative servers for a zone looking for a server which will answer the query.

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

If you support EDNS and get a query with a unsupported EDNS version the correct response is BADVERS [RFC6891].

If you do not support EDNS at all FORMERR and NOTIMP are the expected error codes. That said a minimal EDNS server implementation just requires parsing the OPT records and responding with a empty OPT record. There is no need to interpret any EDNS options present in the request as unsupported options are expected to be ignored [RFC6891].

7. Testing

Verify the server is configured for the zone:

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

expect: status: NOERROR
expect: SOA record

Check that TCP queries work:

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

expect: status: NOERROR
expect: SOA record

Check that queries for a unknown type to work:

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

expect: status: NOERROR
expect: a empty answer section.

Check that queries the CD=1 work:

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

expect: status: NOERROR
expect: SOA record to be present

Check that queries the AD=1 work:

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

expect: status: NOERROR
expect: SOA record to be present

Check that queries with the last unassigned DNS header flag to work:

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

Check that plain EDNS queries work:

dig +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

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

dig +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
(this will change when EDNS version 1 is defined)

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

```
   dig +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
```

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

```
   dig +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
```

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

```
   dig +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
```

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

```
   dig +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
```

   Check that a DNSSEC queries work (EDNS supported):

```
dig +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

    DO=1 as per [RFC3225].

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

```
dig +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

    Check that new opcodes are handled.

```
dig +noedns +noad +opcode=15 +norec soa $zone @$server
```

    expect: status: NOTIMP
    expect: SOA record to not be present

    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.

    It is advisable to run all the above tests in parallel so as to
    minimise the delays due to multiple timeouts when the servers do not
    respond.

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

8. Security Considerations

    Testing protocol compliance can potentially result 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.

9. IANA Considerations

IANA / ICANN needs to consider what tests, if any, from above that it should add to the zone maintenance procedures for zones under its control including pre-delegation checks. Otherwise this document has no actions for IANA.

10. 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>.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <http://www.rfc-editor.org/info/rfc5966>.

[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>.

Author's Address

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

   Email: marka@isc.org