Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: May 29, 2016

M. Andrews
ISC

November 26, 2015

A Common Operational Problem in DNS Servers - Failure To Respond.
draft-andrews-dns-no-response-issue-16

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 classes of queries to which some servers either fail to respond or else respond incorrectly. This document also suggests procedures for TLD and other similar 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

Copyright Notice

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 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 delegations 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, followed by 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 answers 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 should 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 an 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 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. One can construct lists of nameservers from other sources and that has been done to survey the state of the Internet, but that doesn’t give you 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.

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.

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 an 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 extensions that make use of them due to the need to reconfigure / update firewalls.

Requests with unknown EDNS options are not an 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. Whole Answer Caches

Whole answer caches can return the wrong response to a query if they do not take all of the query into account. This has implications when testing and with overall protocol compliance.

e.g. There 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 proceeded by a EDNS version 0 query for the same name and type.

7. 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 an 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 an 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].

8. Testing

Testing is broken into two sections. Basic DNS which all servers should meet and Extended DNS which should be met by all servers that support EDNS.

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

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

8.1. Testing - Basic DNS

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

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
```

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 [RFC5966].

Check that queries for an unknown type to 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]. [RFC5395] 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.

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].

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].

Check that queries with the last unassigned DNS header flag to 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."

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.

8.2. 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.

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.

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].

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].

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].

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.

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.

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].

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.

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
```

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.

9. 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.

10. 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.

11. 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>.

[RFC5395] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", RFC 5395, DOI 10.17487/RFC5395, November 2008, <http://www.rfc-editor.org/info/rfc5395>.

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

[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>.

Author’s Address

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

Email: marka@isc.org

dnsop
Internet-Draft
Intended status: Standards Track
Expires: October 16, 2015

J. Appelbaum
Tor Project Inc.
A. Muffett
Facebook

April 14, 2015

The .onion Special-Use Domain Name
draft-appelbaum-dnsop-onion-tld-01

Abstract

This document registers the ".onion" Special-Use Domain Name.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Tor network [Dingledine2004] has the ability to host network services using the ".onion" Top-Level Domain. Such addresses can be used as other domain names would be (e.g., in URLs [RFC3986]), but instead of using the DNS infrastructure, .onion names functionally correspond to the identity of a given service, thereby combining location and authentication.

In this way, .onion names are "special" in the sense defined by [RFC6761] Section 3; they require hardware and software implementations to change their handling, in order to achieve the desired properties of the name (see Section 4). These differences are listed in Section 2.

Like other TLDs, .onion addresses can have an arbitrary number of subdomain components. This information is not meaningful to the Tor protocol, but can be used in application protocols like HTTP [RFC7230].

See [tor-address] and [tor-rendezvous] for the details of the creation and use of .onion names.

Note that this draft was preceded by [I-D.grothoff-iesg-special-use-p2p-names], which registered .onion alongside other, similar TLDs. Because .onion is in wide use, it has become urgent to expedite its registration. This does not indicate that the other registrations should be abandoned.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The ".onion" Special-Use TLD

These properties have the following effects upon parties using or processing .onion names (as per [RFC6761]):

1. Users: human users are expected to recognize .onion names as having different security properties, and also being only available through software that is aware of onion addresses.

2. Application Software: Applications that implement the Tor protocol MUST recognize .onion names as special by either accessing them directly, or using a proxy (e.g., SOCKS [RFC1928]) to do so. Applications that do not implement the Tor protocol SHOULD generate an error upon the use of .onion, and SHOULD NOT perform a DNS lookup.

3. Name Resolution APIs and Libraries: Resolvers that implement the Tor protocol MUST either respond to requests for .onion names by resolving them (see [tor-rendezvous]) or by responding with NXDOMAIN. Other resolvers SHOULD respond with NXDOMAIN.

4. Caching DNS Servers: Caching servers SHOULD NOT attempt to look up records for .onion names. They SHOULD generate NXDOMAIN for all such queries.

5. Authoritative DNS Servers: Authoritative servers SHOULD respond to queries for .onion with NXDOMAIN.

6. DNS Server Operators: Operators SHOULD NOT configure an authoritative DNS server to answer queries for .onion. If they do so, client software is likely to ignore any results (see above).

7. DNS Registries/Registrars: Registrars MUST NOT register .onion names; all such requests MUST be denied.

3. IANA Considerations

This document registers the "onion" TLD in the registry of Special-Use Domain Names [RFC6761]. See Section 2 for the registration template.

4. Security Considerations

.onion names are often used provide access to end to end encrypted, secure, anonymized services; that is, the identity and location of the server is obscured from the client. The location of the client is obscured from the server. The identity of the client may or may

not be disclosed through an optional cryptographic authentication process.

These properties can be compromised if, for example:

- o The server "leaks" its identity in another way (e.g., in an application-level message), or
- o The access protocol is implemented or deployed incorrectly, or
- o The access protocol itself is found to have a flaw.

.onion names are self-authenticating, in that they are derived from the cryptographic keys used by the server in a client verifiable manner during connection establishment. As a result, the cryptographic label component of a .onion name is not intended to be human-meaningful.

The Tor network is designed to not be subject to any central controlling authorities with regards to routing and service publication, so .onion names cannot be registered, assigned, transferred or revoked. "Ownership" of a .onion name is derived solely from control of a public/private key pair which corresponds to the algorithmic derivation of the name.

Users must take special precautions to ensure that the .onion name they are communicating with is correct, as attackers may be able to find keys which produce service names that are visually or apparently semantically similar to the desired service.

Also, users need to understand the difference between a .onion name used and accessed directly via Tor-capable software, versus .onion subdomains of other TLDs and providers (e.g., the difference between example.onion and example.onion.tld).

The cryptographic label for a .onion name is constructed by applying a function to the public key of the server, the output of which is rendered as a string and concatenated with the string ".onion". Dependent upon the specifics of the function used, an attacker may be able to find a key that produces a collision with the same .onion name with substantially less work than a cryptographic attack on the full strength key. If this is possible the attacker may be able to impersonate the service on the network.

If client software attempts to resolve a .onion name, it can leak the identity of the service that the user is attempting to access to DNS resolvers, authoritative DNS servers, and observers on the

intervening network. This can be mitigated by following the recommendations in Section 2.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

5.2. Informative References

[Dingledine2004]
Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004, <https://www.onion-router.net/Publications/tor-design.pdf>.

[I-D.grothoff-iesg-special-use-p2p-names]
Grothoff, C., Wachs, M., hellekin, h., Appelbaum, J., and L. Ryge, "Special-Use Domain Names of Peer-to-Peer Systems", draft-grothoff-iesg-special-use-p2p-names-04 (work in progress), January 2015.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[tor-address]
Mathewson, N. and R. Dingledine, "Special Hostnames in Tor", September 2001, <https://gitweb.torproject.org/torspec.git/plain/address-spec.txt>.

[tor-rendezvous]
Mathewson, N. and R. Dingledine, "Tor Rendezvous Specification", April 2014, <https://gitweb.torproject.org/torspec.git/plain/rend-spec.txt>.

Appendix A. Acknowledgements

Thanks to Roger Dingledine, Linus Nordberg and Seth David Schoen for their input and review.

This specification builds upon previous work by Christian Grothoff, Matthias Wachs, Hellekin O. Wolf, Jacob Appelbaum and Leif Ryge to register the .onion TLD in conjunction with other, similar TLDs.

Authors' Addresses

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Alec Muffett
Facebook

Email: alecm@fb.com

IETF
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2015

L. Chapin, Ed.
Interisle Consulting Group
M. McFadden, Ed.
InterConnect Communications Ltd

March 2, 2015

Additional Reserved Top Level Domains
draft-chapin-additional-reserved-tlds-02

Abstract

The Internet Domain Name System (DNS) defines a tree of names starting with root, ".", immediately below which are top level domain (TLD) names such as ".com" and ".us". In June 1999 [RFC2606] reserved a small number of TLD names for use in documentation examples, private testing, experiments, and other circumstances in which it is desirable to avoid conflict with current or future actual TLD names in the DNS.

There has been significant evolution of Internet engineering and operation practices since [RFC2606] was published. In February 2013 [RFC6761] defined criteria and procedures for reserving a domain name for special use, and established an IANA registry for such names. This document reserves three domain name labels for special use in accordance with the criteria and procedures of [RFC6761]: home, corp, and mail.

It is important to note that TLD names may be reserved, in other contexts, for policy, political, or other reasons that are distinct from the IETF’s concern with Internet engineering and operations. This document reserves TLD names only for operational and engineering reasons.

Status of This Memo

This Internet-Draft will expire on September 3, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Internet Domain Name System is documented in [RFC1034], [RFC1035], [RFC1591] and numerous additional Requests for Comment. It defines a tree of names starting with root, ".", immediately below which are top level domain names such as ".com" and ".us". Below top level domain names there are normally additional levels of names.

[RFC2606] reserves a small number of TLD names which can be used for private testing of existing DNS related code, examples in documentation, DNS related experimentation, invalid DNS names, or other similar uses without fear of conflicts with current or future actual top-level domain names in the global DNS. [RFC2606] also notes that the Internet Assigned Numbers Authority (IANA) reserves the label "example" at the second level below the TLDs .com, .net, and .org.

Since [RFC2606] was published in 1999, Internet engineering and operation practices have evolved in ways that led to the publication in February 2013 of [RFC6761], which defined criteria and procedures for reserving a domain name for special use and established an IANA registry to which additional reserved special use names might be added as new requirements arose.

This document follows [RFC6761] to add three reserved top-level domain name labels to the IANA special-use names registry. It is prompted by the impending advent of new TLDs which might, in the absence of the reservations for which this document provides, introduce TLD labels that could create engineering and operational problems for root server operators and other DNS infrastructure providers.

It is important to note that TLD names may be reserved, in other contexts, for policy, political, or other reasons that are distinct from the IETF's concern with Internet engineering and operations. This document reserves TLD names only for operational and engineering reasons.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. New top-level domain name reservations

In its report [SAC045] of a quantitative study of queries to the DNS root servers entitled "Invalid Top Level Domain Queries at the Root Level of the Domain Name System" [SAC045] ICANN’s Security and Stability Advisory Committee "calls attention to the potential problems that may arise should a new TLD applicant use a string that has been seen with measurable (and meaningful) frequency in a query for resolution by the root system and the root system has previously generated a response."

Of particular concern is the case in which a string "has been queried and a root name server has responded to the query with a non-existent domain (NXDOMAIN) result, i.e., the string has not been delegated but has been queried." [SAC045] reports the results of a CAIDA measurement study [RSSAC_DNS] which found that "NXDOMAIN responses account for more than 25 percent of the total responses from root name servers observed in the study, and the top ten such strings account for 10 percent of the total query load."

[SAC045] describes in detail the engineering and operational problems that would ensue from the delegation, as new valid TLD names, of previously invalid labels that have frequently appeared in queries to the root: "If the [new TLD label] were to be approved and the TLD included in the root zone, queries to the root level of the DNS for a string that hitherto returned NXDOMAIN would begin to return positive responses containing name servers of the new TLD."

Recommendation (2) of [SAC045] calls for the community to develop principles for "prohibiting the delegation of strings in addition to those already identified in [RFC2606]." As the first step in that process, based on the data reported by [SAC045], this document adds to the list of names that may not be used for top-level domains the following labels:

- home

- corp

These two top-level domain labels are to be added to the "Special-Use Domain Names" registry created by [RFC6761], as described in the IANA Considerations section of this document.

In addition, [SAC062] describes the risks associated with delegating a name in the root of the public DNS that is also used in privately defined namespaces (in which it is also syntactically valid). Users, software, or other functions in the private domain may confuse the private and public instances of the same name. This risk, referred to as "name collision," results in potential harm to enterprise networks that use previously undelegated names at the root of a private namespace when the name is delegated in the public root.

Research conducted by Interisle Consulting Group [INTERISLE] indicates that another name, in addition to those identified by [SAC045], presents a particularly high risk of name collision. This document therefore also adds the following string to the "Special-Use Domain Names" registry:

o mail

Further resesarch, conducted by JAS Advisors on behalf of ICANN [JAS_MITIGATION] shows that the names .corp, .home and .mail are clear and significant risks for name collision. In that report the following recommendation is made: "The TLDs .corp, .home, and .mail be permanently reserved for internal use and receive RFC 1918-like protection/treatment, potentially via RFC 6761."

The three names that are reserved by this document are those on which all three studies (by SSAC, Interisle and JAS Advisors) agree.

4. Security Considerations

The name reservations specified in this document are intended to reduce the risk of harmful collision between names that are in well-established common use as TLDs in private namespaces and syntactically identical names that could otherwise be delegated as TLDs in the global DNS.

The security concerns associated with name collision are well presented in [SAC045], [SAC062], the Interisle report [INTERISLE], and the ICANN report "Name Collision Identification and Mitigation for IT Professionals" [ICANN_MITIGATION].

5. IANA Considerations

This document specifies three new labels to be added to the "Special-Use Domain Names" registry maintained by IANA pursuant to [RFC6761]. The labels are to be added to the registry in the following way:

Name	Reference
home	[RFC-to-be]
corp	[RFC-to-be]
mail	[RFC-to-be]

Figure 1

5.1. Domain Name Reservation Considerations for home

5.1.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.1.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.1.3. Name Resolution APSs and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.1.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn’t prevent them from using this reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these

names resolved locally rather than on the public Internet. Compliant name server software MUST NOT reject these names as invalid. Instead, name server software SHOULD allow for local resolution of the name and SHOULD NOT transmit a query for resolution into the public DNS.

5.1.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document MUST be denied.

5.2. Domain Name Reservation Considerations for corp

5.2.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.2.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.2.3. Name Resolution APSs and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.2.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn’t prevent them from using this reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always

treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these names resolved locally rather than on the public Internet. Compliant name server software MUST NOT reject these names as invalid. Instead, name server software SHOULD allow for local resolution of the name and SHOULD NOT transmit a query for resolution into the public DNS.

5.2.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document MUST be denied.

5.3. Domain Name Reservation Considerations for mail

5.3.1. Users

Are human users expected to recognize these names as special and use them differently? In what way?

The reservations provided in this document are intended to reduce spurious queries at the root of the DNS and avoid potential collisions between resolutions of names in private name spaces and the public DNS. Users do not have to know that these names are special.

5.3.2. Application Software

Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

These names are being added to the Special-Use Domain Name registry, in part, because some application software implementations have long used these names for special purposes in private networks. Developers of new applications do not need to filter or test for the names. Instead, the intent is to reserve the names for local use and avoid unnecessary queries in the public DNS.

5.3.3. Name Resolution APSs and Libraries

Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

Authors of name resolution APIs and libraries SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.4. Caching DNS Servers

Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of caching domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.5. Authoritative DNS Servers

Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

Authors of authoritative domain name server software SHOULD restrict these names to local resolution and SHOULD NOT allow queries for strings that use these Special-Use Domain Names to be forwarded to the public DNS for resolution.

5.3.6. DNS Server Operators

Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn’t prevent them from using this

reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

The intent of the reservations in this IANA Considerations section is to prevent spurious and potentially problematic queries from appearing in the public DNS. DNS server operators SHOULD always treat strings with the Special-Use Domain Names in section 5 as names for local resolution.

Since these strings are intended to have local use, it is quite possible that DNS operators would configure an authoritative DNS server as authoritative for these reserved names in a private network. This would be consistent with the goal of having these names resolved locally rather than on the public Internet. Compliant name server software MUST NOT reject these names as invalid. Instead, name server software SHOULD allow for local resolution of the name and SHOULD NOT transmit a query for resolution into the public DNS.

5.3.7. DNS Registries/Registrars

How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially-designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a web site at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Requests to register any names added to the Special-Use Domain Name registry as part of the IANA Considerations section of this document MUST be denied.

6. References

7. Acknowledgments

8. References

8.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1591] Postel, J., "Domain Name System Structure and Delegation", RFC 1591, March 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

8.2. Informative References

[ICANN_MITIGATION]
Internet Corporation for Assigned Names and Numbers, "Guide to Name Collision Identification and Mitigation for IT Professionals", August 2013, <http://www.icann.org/en/about/staff/security/ssr/name-collision-mitigation-05dec13-en.pdf>.

[INTERISLE]
Chapin, L., "Name Collision in the DNS", August 2013, <http://www.icann.org/en/about/staff/security/ssr/name-collision-02aug13-en.pdf>.

[JAS_MITIGATION]
Internet Corporation for Assigned Names and Numbers, "Mitigating the Risk of DNS Namespace Collisions", February 2014, <https://www.icann.org/en/about/staff/security/ssr/name-collision-mitigation-26feb14-en.pdf>.

[RSSAC_DNS]
claffy, kc., "DNS Research Update from CAIDA Status and Recent Experiences", March 2009, <http://www.caida.org/publications/presentations/2009/rssac_dns/rssac_dns.pdf>.

[SAC045] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System", December 2010, <http://www.icann.org/en/committees/security/sac045.pdf>.

[SAC062] ICANN Security and Stability Advisory Committee, "SSAC Advisory Concerning the Mitigation of Name Collision Risk", November 2013, <http://www.icann.org/en/groups/ssac/documents/sac-062-en.pdf>.

Authors' Addresses

Lyman Chapin (editor)
Interisle Consulting Group
125A Magazine Street
Cambridge, MA 02139
UK

Phone: +1 617 686 2527
Email: lyman@interisle.net

Mark McFadden (editor)
InterConnect Communications Ltd
Merlin House; Station Road
Chepstow, Monmouthshire NP16 5PB
UK

Phone: +44 7792 276 904
Email: markmcfadden@icc-uk.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 19, 2016

K. Fujiwara
JPRS
A. Kato
Keio/WIDE

March 18, 2016

Aggressive use of NSEC/NSEC3
draft-fujiwara-dnsop-nsec-aggressiveuse-03

Abstract

While DNS highly depends on cache, its cache usage of non-existence information has been limited to exact matching. This draft proposes the aggressive use of a NSEC/NSEC3 resource record, which is able to express non-existence of a range of names authoritatively. With this proposal, it is expected that shorter latency to many of negative responses as well as some level of mitigation of random sub-domain attacks (referred to as "Water Torture" attacks). It is also expected that non-existent TLD queries to Root DNS servers will decrease.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

While negative (non-existence) information of DNS caching mechanism has been known as DNS negative cache [RFC2308], it requires exact matching in most cases. Assume that "example.com" zone doesn't have names such as "a.example.com" and "b.example.com". When a full-service resolver receives a query "a.example.com" , it performs a DNS

resolution process, and eventually gets NXDOMAIN and stores it into its negative cache. When the full-service resolver receives another query "b.example.com", it doesn’t match with "a.example.com". So it will send a query to one of the authoritative servers of "example.com". This was because the NXDOMAIN response just says there is no such name "a.example.com" and it doesn’t tell anything for "b.example.com".

Section 5 of [RFC2308] seems to show that negative answers should be cached only for the exact query name, and not (necessarily) for anything below it.

Recently, DNSSEC [RFC4035] [RFC5155] has been practically deployed. Two types of resource record (NSEC and NSEC3) along with their RRSIG records represent authentic non-existence. For a zone signed with NSEC, it would be possible to use the information carried in NSEC resource records to indicate that a range of names where no valid name exists. Such use is discouraged by Section 4.5 of RFC 4035, however.

This document proposes to make a minor change to RFC 4035 and a full-service resolver can use NSEC/NSEC3 resource records aggressively so that the resolver responds with NXDOMAIN immediately if the name in question falls into a range expressed by a NSEC/NSEC3 resource record.

Aggressive Negative Caching was first proposed in Section 6 of DNSSEC Lookaside Validation (DLV) [RFC5074] in order to find covering NSEC records efficiently. Unbound [UNBOUND] has aggressive negative caching code in its DLV validator. Unbound TODO file contains "NSEC/NSEC3 aggressive negative caching".

Section 3 of [I-D.vixie-dnsext-resimprove] ("Stopping Downward Cache Search on NXDOMAIN") proposed another approach to use NXDOMAIN information effectively.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Many of the specialized terms used in this specification are defined in DNS Terminology [RFC7719].

3. Problem Statement

Random sub-domain attacks (referred to as "Water Torture" attacks or NXDomain attacks) send many non-existent queries to full-service resolvers. Their query names consist of random prefixes and a target domain name. The negative cache does not work well and target full-service resolvers result in sending queries to authoritative DNS servers of the target domain name.

When number of queries is large, the full-service resolvers drop queries from both legitimate users and attackers as their outstanding queues are filled up.

For example, BIND 9.10.2 [BIND9] full-service resolvers answer SERVFAIL and Unbound 1.5.2 full-service resolvers drop most of queries under 10,000 queries per second attack.

The countermeasures implemented at this moment are rate limiting and disabling name resolution of target domain names in ad-hoc manner.

4. Proposed Solution

4.1. Aggressive Negative Caching

If the target domain names are DNSSEC signed, aggressive use of NSEC/NSEC3 resource records mitigates the problem.

Section 4.5 of [RFC4035] shows that "In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace".

To reduce non-existent queries sent to authoritative DNS servers, it is suggested to relax this restriction as follows:

```
+-------------------------------------------------------------+
|  DNSSEC enabled full-service resolvers MAY use               |
|  NSEC/NSEC3 resource records to generate negative responses  |
|  until their effective TTLs or signatures on the records     |
|  in question expire.                                         |
+-------------------------------------------------------------+
```

If the full-service resolver’s cache have enough information to validate the query, the full-service resolver MAY use NSEC/NSEC3/

wildcard records aggressively. Otherwise, the full-service resolver MUST fall back to send the query to the authoritative DNS servers.

Necessary information to validate are matching/covering NSEC/NSEC3 of the wildcards which may match the query name, matching/covering NSEC/NSEC3 of non-terminals which derive from the query name and matching/covering NSEC/NSEC3 of the query name.

If the query name has the matching NSEC/NSEC3 RR and it shows the query type does not exist, the full-service resolver is possible to respond with NODATA (empty) answer.

4.2. NSEC

A full-service resolver implementation SHOULD support aggressive use of NSEC and enable it by default. It SHOULD provide a configuration knob to disable aggressive use of NSEC.

The validating resolver need to check the existence of matching wildcards which derive from the query name, covering NSEC RRs of the matching wildcards and covering NSEC RR of the query name.

If the full-service resolver’s cache contains covering NSEC RRs of matching wildcards and the covering NSEC RR of the query name, the full-service resolver is possible to respond with NXDOMAIN error immediately.

4.3. NSEC3

NSEC3 aggressive negative caching is more difficult. If the zone is signed with NSEC3, the validating resolver need to check the existence of non-terminals and wildcards which derive from query names.

If the full-service resolver’s cache contains covering NSEC3 RRs of matching wildcards, the covering NSEC3 RRs of the non-terminals and the covering NSEC3 RR of the query name, the full-service resolver is possible to respond with NXDOMAIN error immediately.

If the validating resolver proves the non-exisence of the non-terminal domain name of the query name, the query name does not exist.

To identify signing types of the zone, validating resolvers need to build separated cache of NSEC and NSEC3 resource records for each signer domain name.

When a query name is not in the regular cache, find closest enclosing NS RRset in the regular cache. The owner of the closest enclosing NS RRset may be the longest signer domain name of the query name. If there is no entry in the NSEC/NSEC3 cache of the signer domain name, aggressive negative caching is not possible at this moment. Otherwise, there is at least one NSEC or NSEC3 resource records. The record shows the signing type.

A full-service resolver implementation MAY support aggressive use of NSEC3. It SHOULD provide a configuration knob to disable aggressive use NSEC3 in this case.

4.4. NSEC3 Opt-Out

If the zone is signed with NSEC3 and with Opt-Out flag set to 1, the aggressive negative caching is not possible at the zone.

4.5. Wildcard

Even if a wildcard is cached, it is necessary to send a query to an authoritative server to ensure that the name in question doesn’t exist as long as the name is not in the negative cache.

When aggressive use is enabled, regardless of description of Section 4.5 of [RFC4035], it is possible to send a positive response immediately when the name in question matches a NSEC/NSEC3 RRs in the negative cache.

4.6. Consideration on TTL

This function needs care on the TTL value of negative information because newly added domain names cannot be used while the negative information is effective. RFC 2308 states the maximum number of negative cache TTL value is 10800 (3 hours). So the full-service resolver SHOULD limit the maximum effective TTL value of negative responses (NSEC/NSEC3 RRs) to 10800 (3 hours). It is reasonably small but still effective for the purpose of this document as it can eliminate significant amount of DNS attacks with randomly generated names.

5. Additional Considerations

5.1. The CD Bit

The CD bit disables signature validation. It is one of the basic functions of DNSSEC protocol and it SHOULD NOT be changed. However, attackers may set the CD bit to their attack queries and the aggressive negative caching will be of no use.

Ignoring the CD bit function may break the DNSSEC protocol.

This draft proposes that the CD bit may be ignored to support aggressive negative caching when the full-service resolver is under attacks with CD bit set.

5.2. Detecting random subdomain attacks

Full-service resolvers should detect conditions under random subdomain attacks. When they are under attacks, their outstanding queries increase. If there are some destination addresses whose outstanding queries are many, they may contain attack target domain names. Existing countermeasures may implement attack detection.

6. Possible side effect

Aggressive use of NSEC/NSEC3 resource records may decrease queries to Root DNS servers.

People may generate many typos in TLD, and they will result in unnecessary DNS queries. Some implementations leak non-existent TLD queries whose second level domain are different each other. Well observed TLDs are ".local" and ".belkin". With this proposal, it is possible to return NXDOMAIN immediately to such queries without further DNS recursive resolution process. It may reduces round trip time, as well as reduces the DNS queries to corresponding authoritative servers, including Root DNS servers.

7. Additional proposals

There are additional proposals to the aggressive negative caching.

7.1. Partial implementation

It is possible to implement aggressive negative caching partially.

DLV aggressive negative caching [RFC5074] is an implementation of NSEC aggressive negative caching which targets DLV domain names.

NSEC only aggressive negative caching is easier to implement NSEC/NSEC3 aggressive negative caching (full implantation) because NSEC3 handling is hard to implement.

Root only aggressive negative caching is possible. It uses NSEC and RRSIG resource records whose signer domain name is root.

An implementation without detecting attacks is possible. It cannot ignore the CD bit and the effectiveness may be limited.

7.2. Aggressive negative caching without DNSSEC validation

Aggressive negative caching may be applicable to full-service resolvers without DNSSEC validation. They can set DNSSEC OK bit in query packets to obtain corresponding NSEC/NSEC3 resource records. While the full-service resolvers SHOULD validate the NSEC/NSEC3 resource records, they MAY use the records to respond NXDOMAIN error immediately without DNSSEC validation.

However, it is highly recommended to apply DNSSEC validation.

7.3. Aggressive negative caching flag idea

Authoritative DNS servers that dynamically generate NSEC records normally generate minimally covering NSEC Records [RFC4470]. Aggressive negative caching does not work with minimally covering NSEC records. Most of DNS operators don’t want zone enumeration and zone information leaks. They prefer NSEC resource records with narrow ranges. When there is a flag that show a full-service resolver support the aggressive negative caching and a query have the aggressive negative caching flag, authoritative DNS servers can generate NSEC resource records with wider range under random subdomain attacks.

However, changing range of minimally covering NSEC Records may be implemented by detecting attacks. Authoritative DNS servers can answer any range of minimally covering NSEC Records.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

Newly registered resource records may not be used immediately. However, choosing suitable TTL value will mitigate the problem and it is not a security problem.

It is also suggested to limit the maximum TTL value of NSEC resource records in the negative cache to, for example, 10800 seconds (3hrs), to mitigate the issue. Implementations which comply with this proposal is suggested to have a configurable maximum value of NSEC RRs in the negative cache.

Aggressive use of NSEC/NSEC3 resource records without DNSSEC validation may cause security problems.

10. Implementation Status

Unbound has aggressive negative caching code in its DLV validator. The author implemented NSEC aggressive caching using Unbound and its DLV validator code.

11. Acknowledgments

The authors gratefully acknowledge DLV [RFC5074] author Samuel Weiler and Unbound developers. Olafur Gudmundsson and Pieter Lexis proposed aggressive negative caching flag idea. Valuable comments were provided by Bob Harold, Tatuya JINMEI, Shumon Huque, Mark Andrews, and Casey Deccio.

12. Change History

This section is used for tracking the update of this document. Will be removed after finalize.

12.1. Version 01

- o Added reference to DLV [RFC5074] and imported some sentences.
- o Added Aggressive Negative Caching Flag idea.
- o Added detailed algorithms.

12.2. Version 02

- o Added reference to [I-D.vixie-dnsext-resimprove]
- o Added considerations for the CD bit
- o Updated detailed algorithms.
- o Moved Aggressive Negative Caching Flag idea into Additional Proposals

12.3. Version 03

- o Added "Partial implementation"
- o Section 4,5,6 reorganized for better representation
- o Added NODATA answer in Section 4
- o Trivial updates

- o Updated pseudo code

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

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

[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>.

[RFC4470] Weiler, S. and J. Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing", RFC 4470, DOI 10.17487/RFC4470, April 2006, <http://www.rfc-editor.org/info/rfc4470>.

[RFC5074] Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074, DOI 10.17487/RFC5074, November 2007, <http://www.rfc-editor.org/info/rfc5074>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <http://www.rfc-editor.org/info/rfc5155>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

13.2. Informative References

[BIND9] Internet Systems Consortium, Inc., "Name Server Software", 2000, <https://www.isc.org/downloads/bind/>.

[I-D.vixie-dnsext-resimprove] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsext-resimprove-00 (work in progress), June 2010.

[UNBOUND] NLnet Labs, "Unbound DNS validating resolver", 2006, <http://www.unbound.net/>.

Appendix A. Aggressive negative caching from RFC 5074

Imported from Section 6 of [RFC5074].

Previously, cached negative responses were indexed by QNAME, QCLASS, QTYPE, and the setting of the CD bit (see RFC 4035, Section 4.7), and only queries matching the index key would be answered from the cache. With aggressive negative caching, the validator, in addition to checking to see if the answer is in its cache before sending a query, checks to see whether any cached and validated NSEC record denies the existence of the sought record(s).

Using aggressive negative caching, a validator will not make queries for any name covered by a cached and validated NSEC record. Furthermore, a validator answering queries from clients will synthesize a negative answer whenever it has an applicable validated NSEC in its cache unless the CD bit was set on the incoming query.

Imported from Section 6.1 of [RFC5074].

Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC records in order to efficiently find covering NSEC records. Only NSEC records from DLV domains need to be included in this data structure.

Appendix B. Detailed implementation idea

Section 6.1 of [RFC5074] is expanded as follows.

Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC and NSEC3 records for each signer domain name of NSEC / NSEC3 records in order to efficiently find covering NSEC / NSEC3 records. Call the table as NSEC_TABLE.

The aggressive negative caching may be inserted at the cache lookup part of the full-service resolvers.

If errors happen in aggressive negative caching algorithm, resolvers MUST fall back to resolve the query as usual. "Resolve the query as usual" means that the full-resolver resolve the query in Recursive-mode as if the full-service resolver does not implement aggressive negative caching.

To implement aggressive negative caching, resolver algorithm near cache lookup will be changed as follows:

```
QNAME = the query name;
QTYPE = the query type;
if ({QNAME,QTYPE} entry exists in the cache) {
    // the resolver responds the RRSet from the cache
    resolve the query as usual;
}

// if NSEC* exists, QTYPE existence is proved by type bitmap
if (matching NSEC/NSEC3 of QNAME exists in the cache) {
    if (QTYPE exists in type bitmap of NSEC/NSEC3 of QNAME) {
        // the entry exists, however, it is not in the cache.
        // need to iterate QNAME/QTYPE.
        resolve the query as usual;
    } else {
        // QNAME exists, QTYPE does not exist.
        the resolver can generate NODATA response;
    }
}

// Find closest enclosing NS RRset in the cache.
// The owner of this NS RRset will be a suffix of the QNAME
//    - the longest suffix of any NS RRset in the cache.
SIGNER = closest enclosing NS RRSet of QNAME in the cache;

// Check the SOA RR of the SIGNER
if (SOA RR of SIGNER does not exist in the cache
    or SIGNER zone is not signed or not validated) {
   Resolve the query as usual;
}

if (SIGNER zone does not have NSEC_TABLE) {
    Resolve the query as usual;
}

if (SIGNER zone is signed with NSEC) { // NSEC mode

    // Check the non-existence of QNAME
    CoveringNSEC = Find the covering NSEC of QNAME;
    if (Covering NSEC doesn't exist in the cache) {
        Resolve the query as usual.
    }

    // Select the longest existing name of QNAME from covering NSEC
    LongestExistName = common part of both owner name and
                       next domain name of CoveringNSEC;
```

```
        if (*.LongestExistName entry exists in the cache) {
            the resolver can generate positive response
            // synthesize the wildcard *.TEST
        }
        if covering NSEC RR of "*.LongestExistName" at SIGNER zone exists
             in the cache {
            the resolver can generate negative response;
        }
        //*.LongestExistName may exist. cannot generate negative response
        Resolve the query as usual.

    } else
    if (SIGNER zone is signed with NSEC3 and does not use Opt-Out) {
        // NSEC3 mode

        TEST = SIGNER;
        while (TEST != QNAME) {
            // if any error happens in this loop, break this loop
            UPPER = TEST;
            add a label from the QNAME to the start of TEST;
              // TEST = label.UPPER
            if (TEST name entry exist in the cache
                || matching NSEC3 of TEST exist in the cache) {
                // TEST exist
                continue; // need to check rest of QNAME
            }
            if (covering NSEC3 of TEST exist in the cache) {
                // (non-)terminal name TEST does not exist
                if (*.UPPER name entry exist in the cache) {
                    // TEST does not exist and *.UPPER exist
                    the resolver can generate positive response;
                } else
                if (covering NSEC3 of *.UPPER exist in the cache) {
                    // TEST does not exist and *.UPPER does not exist
                    the resolver can generate negative response;
                }
                break; // Lack of information (No *.UPPER information)
            }
            break; // Lack of information (No TEST information)
        }
        // no matching/covering NSEC3 of QNAME information
        Resolve the query as usual
    }
```

Authors' Addresses

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Akira Kato
Keio University/WIDE Project
Graduate School of Media Design, 4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
Japan

Phone: +81 45 564 2490
Email: kato@wide.ad.jp

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 7, 2015

P. Hoffman
VPN Consortium
A. Sullivan
Dyn
K. Fujiwara
JPRS
March 06, 2015

DNS Terminology
draft-hoffman-dns-terminology-02

Abstract

The DNS is defined in literally dozens of different RFCs. The terminology used in by implementers and developers of DNS protocols, and by operators of DNS systems, has sometimes changed in the decades since the DNS was first defined. This document gives current definitions for many of the terms used in the DNS in a single document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The domain name system (DNS) is a simple query-response protocol whose messages in both directions have the same format. The protocol and message format are defined in [RFC1034] and [RFC1035]. These RFCs defined some terms, but later documents defined others. Some of the terms from RFCs 1034 and 1035 now have somewhat different meanings than they did in 1987.

This document collects a wide variety of DNS-related terms. Some of them have been precisely defined in earlier RFCs, some have been loosely defined in earlier RFCs, and some are not defined in any earlier RFC at all.

The definitions here are believed to be the consensus definition of the DNS community, both protocol developers and operators. Some of the definitions differ from earlier RFCs, and those differences are noted. The terms are organized loosely by topic. Some definitions are for new terms for things that are commonly talked about in the DNS community but that never had terms defined for them.

In this document, where the consensus definition is the same as the one in an RFC, that RFC is quoted. Where the consensus definition

has changed somewhat, the RFC is mentioned but the new stand-alone definition is given.

Other organizations sometimes define DNS-related terms their own way. For example, the W3C defines "domain" at https://specs.webplatform.org/url/webspecs/develop/.

Note that there is no single consistent definition of "the DNS". It can be considered to be some combination of the following: a commonly-used naming scheme for objects on the Internet; a database representing the names and certain properties of these objects; an architecture providing distributed maintenance, resilience, and loose coherency for this database; and a simple query-response protocol (as mentioned in the current draft) implementing this architecture.

Capitalization in DNS terms is often inconsistent between RFCs and between DNS practitioners. The capitalization used in this document is a best guess at current practices, and is not meant to indicate that other capitalization styles are wrong or archaic.

2. Names

Domain name -- Section 3.1 of RFC 1034 talks of "the domain name space" as a tree structure. "Each node has a label, which is zero to 63 octets in length. ... The domain name of a node is the list of the labels on the path from the node to the root of the tree. ... To simplify implementations, the total number of octets that represent a domain name (i.e., the sum of all label octets and label lengths) is limited to 255."

Fully-qualified domain name (FQDN) -- This is often just a clear way of saying the same thing as "domain name of a node", as outlined above. However, the term is ambiguous. Strictly speaking, a fully-qualified name would include every label, including the final, zero-length label of the root zone: such a name would be written "www.example.net." (note the terminating dot). But because every name eventually shares the common root, names are often written relative to the root (such as "www.example.net") and are still called "fully qualified".
This term first appeared in [RFC1206].

Host name -- This term and its equivalent, "hostname", have been widely used but are not defined in RFC 1034, 1035, 1123, or 2181. The DNS was originally deployed into the Host Tables environment as outlined in [RFC0952], and it is likely that the term followed informally from the definition there. Over time, the definition seems to have shifted. "Host name" is often meant to be a domain name that follows the rules in Section 3.5 of RFC 1034, the

"preferred name syntax". Note that any label in any domain name can contain any octet value; hostnames are generally considered to be domain names where every label follows the rules in the "preferred name syntax", with the amendment that labels can start with ASCII digits (this amendment comes from Section 2.1 of [RFC1123]).

People also sometimes use the term hostname to refer to just the first label of an FQDN. In addition, people sometimes use this term to describe any name that refers to a machine, and those might include labels that do not conform to the "preferred name syntax".

TLD -- A Top-Level Domain, meaning a zone that is one layer below the root, such as .com or .jp. There is nothing special, from the point of view of the DNS, about TLDs. Most of them are also delegation-centric zones, and there are significant policy issues around their operation.

ccTLD -- A TLD that is allocated to a country. Historically, these were two-letter TLDs, and were allocated to countries using the two-letter code from the ISO 3166-1 alpha-2 standard [ISO3166]. In recent years, there have been allocations of TLDs that conform to IDNA2008 ([RFC5890], [RFC5891], [RFC5892], [RFC5893], and [RFC5894]); these are still treated as ccTLDs for policy purposes.

gTLD -- A "generic" TLD is a TLD that is not a ccTLD, and is not one of the small number of historical TLDs such as .int and .arpa. There is no precise definition for which TLDs that are not ccTLDs are gTLDs.

3. DNS Message Format

Header -- The first 12 octets of a DNS message. Many of the fields and flags in the header diagram in section 4.1.1 of RFC 1035 are referred to by their names in that diagram. For example, the response codes are called "RCODEs", and the authoritative answer bit is often called "the AA flag" or "the AA bit". RCODEs are covered in Section 4.

TTL -- The maximum "time to live" of a resource record. A TTL value is an unsigned number, with a minimum value of 0, and a maximum value of 2147483647. That is, a maximum of 2^31 - 1. When transmitted, the TTL is encoded in the less significant 31 bits of the 32 bit TTL field, with the most significant, or sign, bit set to zero. (Quoted from [RFC2181], section 8) (Note that RFC 1035 erroneously stated that this is a signed integer; it is fixed in an erratum.)

The TTL "specifies the time interval that the resource record may be cached before the source of the information should again be

consulted". (Quoted from RFC 1035, section 3.2.1) Also: "the time interval (in seconds) that the resource record may be cached before it should be discarded". (Quoted from RFC 1035, section 4.1.3). Despite being defined for a resource record, the TTL of every resource record in an RRset is required to be the same (RFC2181, section 5.2).

The reason that the TTL is the maximum time to live is that a cache operator might decide to shorten the time to live for operational purposes, such as if there is a policy to not allow TTL values over a certain number. Also, if a value is flushed from the cache when its value is still positive, the value effectively becomes zero.

There is also the concept of a "default TTL" for a zone, which can be a configuration parameter in the server software. This is often expressed by a default for the entire server, and an default override for a zone using the $TTL directive in a zone file. The $TTL directive was added to the master file format by [RFC2308].

Glue records -- Resource records which are not part of the authoritative data, and are address resource records for the servers listed in the message. They contain data that allows access to name servers for subzones. (Definition from RFC 1034, section 4.2.1)

Referrals -- Data from the authority section of a non-authoritative answer. RFC 1035 section 2.1 defines "authoritative" data. However, referrals at zone cuts are not authoritative. Referrals may be a zone cut NS resource records and their glue. NS records on the parent side of a zone cut are an authoritative delegation, but are not treated as authoritative data by the client. [[A more complete and precise definition will be needed here.]]

4. Response Codes

Some of response codes that are defined in RFC 1035 have gotten their own shorthand names. Some common response code (RCODE) names that appear without reference to the numeric value are "FORMERR", "SERVFAIL", and "NXDOMAIN". All of the RCODEs are listed at http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml, although that site uses mixed-case capitalization, while most documents use all-caps.

NODATA -- This is not an actual response code, but instead is the combination of an RCODE of 0 (NOERROR) and an Answer section that is empty. That is, it indicates that the response is no answer, but that there was not supposed to be one. Section 1 of RFC 2308 defines it as "a pseudo RCODE which indicates that the name is valid, for the given class, but are no records of the given type."

5. Resource Records

RR -- A short form for resource record. (RFC 1034, section 3.6.)

RRset -- A set of resource records with the same label, class and type, but with different data. (Definition from RFC 2181) Also spelled RRSet in some documents. As a clarification, "same label" in this definition means "same owner name".

OPT -- A pseudo-RR (sometimes called a meta-RR) that is used only to contain control information pertaining to the question-and-answer sequence of a specific transaction. (Definition from [RFC6891], section 6.1.1)

Owner -- The domain name where a RR is found (RFC 1034, section 3.6). Often appears in the term "owner name".

SOA field names -- DNS documents, including the definitions here, often refer to the fields in the RDATA an SOA resource record by field name. Those fields are defined in Section 3.3.13 of RFC 1035. The names (in the order they appear in the SOA RDATA) are MNAME, RNAME, SERIAL, REFRESH, RETRY, and EXPIRE, MINIMUM. Note that the meaning of MINIMUM field is updated in Section 4 of RFC 2308; the new definition is that the MINIMUM field is only "the TTL to be used for negative responses".

6. DNS Servers

This section defines the terms used for the systems that act as DNS clients, DNS servers, or both. Some terms about servers describe servers that do and do not use DNSSEC; see Section 9 for those definitions.

[[There is a request to "first describe the iterative and recursive resolution processes, and mention the expected values of the RD,RA,AA bits. Then you can describe the distinctions between recursive and iterative clients, and between recursive and authoritative servers, in terms of the roles they play in the different resolution processes." This would require the section to be quite different than the other sections in the document.]]

Resolver -- A program that extracts information from name servers in response to client requests. (Quoted from RFC 1034, section 2.4) It is a program that interfaces user programs to domain name servers. The resolver is located on the same machine as the program that requests the resolver’s services. (Quoted from RFC 1034, section 5.1) A resolver performs queries for a name, type, and class, and receives answers. The logical function is called "resolution". In

practice, the term is usually referring to some specific type of resolver (some of which are defined below), and understanding the use of the term depends on understanding the context.

Stub resolver -- A resolver that cannot perform all resolution itself. Stub resolvers generally depend on a recursive resolver to undertake the actual resolution function. Stub resolvers are discussed but never fully defined in RFC 1034, section 5.3.1.

Iterative mode -- A resolution mode of a server that receives DNS queries and responds with a referral to another server. Section 2.3 of RFC 1034 describes this as "The server refers the client to another server and lets the client pursue the query". A resolver that works in iterative mode is sometimes called an "iterative resolver".

Recursive mode -- A resolution mode of a server that receives DNS queries and either responds to those queries from a local cache or sends queries to other servers in order to get the final answers to the original queries. Section 2.3 of RFC 1034 describes this as "The first server pursues the query for the client at another server". A server operating in recursive mode may be thought of as having a name server side (which is what answers the query) and a resolver side (which performs the resolution function). Systems operating in this mode are commonly called "recursive servers". Sometimes they are called "recursive resolvers". While strictly the difference between these is that one of them sends queries to another recursive server and the other does not, in practice it is not possible to know in advance whether the server that one is querying will also perform recursion; both terms can be observed in use interchangeably.

Priming -- The mechanism used by a resolvrer to determine where to send queries before there is anything in the resolver’s cache. Priming is most often done from a configuration setting that contains a list of authoritative servers for the DNS root zone.

Negative caching -- The storage of knowledge that something does not exist, cannot give an answer, or does not give an answer. (Quoted from Section 1 of RFC 2308)

Authoritative server -- A system that responds to DNS queries with information about zones for which it has been configured to answer with the AA flag in the response header set to 1. It is a server that has authority over one or more DNS zones. Note that it is possible for an authoritative server to respond to a query without the parent zone delegating authority to that server.

Slave -- An authoritative server which uses zone transfer to retrieve the zone. (Quoted from [RFC1996], section 2.1)

Master -- Any authoritative server configured to be the source of zone transfer for one or more slave servers. (Quoted from RFC 1996, section 2.1)

Primary master -- The primary master is named in the zone’s SOA MNAME field and optionally by an NS resource record. (Quoted from RFC 1996, section 2.1)

Stealth server -- This is the same as a slave server except that it is not listed in an NS resource record for the zone. (Quoted from RFC 1996, section 2.1) A stealth server is often actually a master for zone transfers, and in that case is called a "hidden master".

Zone transfer -- The act of a client requesting a copy of a zone and an authoritative server sending the needed information. There are two common standard ways to do zone transfers: the AXFR ("Authoritative Transfer") mechanism to copy the full zone, and the IXFR ("Incremental Transfer") mechanism to copy only parts of the zone that have changed. Many systems use non-standard methods for zone transfer outside the DNS protocol.

DNS forwarder -- A system that receives a DNS query, possibly changes the query, sends the resulting query to a recursive resolver, receives the response from a resolver, possibly changes the response, and sends the resulting response to the stub resolver. Section 1 of RFC 2308 describes a forwarder as "a nameserver used to resolve queries instead of directly using the authoritative nameserver chain". RFC further says "The forwarder typically either has better access to the internet, or maintains a bigger cache which may be shared amongst many resolvers."

[RFC5625] does not give a specific definition for DNS forwarder, but describes in detail what features they need to support. The protocol interfaces for DNS forwarders are exactly the same as those for recursive resolvers (for interactions with DNS stubs) and as those for stub resolvers (for interactions with recursive resolvers).

Full resolver -- This term is used in RFC 1035, but it is not defined there. RFC 1123 defines a "full-service resolver" that may or may not be what was intended by "full resolver" in RFC 1035. In the vernacular, a full-service resolver is usually one that would be suitable for use by a stub resolver.

Consensual policy-implementing resolver -- A resolver that changes some answers it returns based on policy criteria, such as to prevent

access to malware sites. These policy criteria are agreed to by systems that query this resolver through some out of band mechanism (such as finding out about the resolver from a web site and reading the policy).

Non-consensual policy-implementing resolver -- A resolver that is not a consensual policy-implementing resolver that changes the answers it returns. The difference between this and a consensual policy-implementing resolver is that users of this resolver are not expected to know that there is a policy to change the answers it returns.

Open resolver -- A full resolver that accepts and processes queries from any (or nearly any) stub resolver. This is sometimes also called a "public resolver".

Open forwarder -- A DNS forwarder that accepts and forwards queries from any (or nearly any) stub resolver to a full resolver.

Views -- A view is a configuration for a server that allows it to provide different answers depending on the address on the query. Views are often used to provide more names or different addresses to queries from "inside" a protected network than to those "outside" that network. Views are not a standardized part of the DNS, but they are widely implemented in server software.

Passive DNS -- A mechanism to collect large amounts of DNS data by storing queries and responses from many recursive resolvers. Passive DNS databases can be used to answer historical questions about DNS zones such as which records were available for them at what times in the past.

Child-centric resolver -- A DNS resolver that, instead of serving the NS RRset and glue records that it obtained from the parent of a zone, serves data from the authoritative servers for that zone. The term "child-centric" is meant as the opposite of "parent-centric", which means a resolver that simply serves the NS RRset and glue records for a zone that it obtained from the zone’s parent, without checking the authoritative servers for that zone.

7. Zones

This section defines terms that are used when discussing zones that are being served or retrieved.

Zone -- A unit of organization of authoritative data. Zones can be automatically distributed to the name servers which provide redundant service for the data in a zone. (Quoted from RFC 1034, section 2.4).

Child -- The entity on record that has the delegation of the domain from the Parent. (Quoted from [RFC7344], section 1.1)

Parent -- The domain in which the Child is registered. (Quoted from RFC 7344, section 1.1) Earlier, "parent name server" was defined in [RFC0882] as "the name server that has authority over the place in the domain name space that will hold the new domain".

Origin -- 1. The domain name that appears at the top of a zone. 2. The domain name within which a given relative domain name appears in zone files. Generally seen in the context of "$ORIGIN", which is a control entry defined in RFC 1035, section 5.1, as part of the master file format. For example, if the $ORIGIN is set to "example.org.", then a master file line for "www" is in fact an entry for "www.example.org.".

Zone cut -- The delimitation point between two zones where the origin of one of the zones is the child of the other zone. (Section 6 of RFC 2181 uses this term extensively, although never actually defines it.) Section 4.2 of RFC 1034 uses "cuts" as "zone cut".

Apex -- The point in the tree at an owner of an SOA and corresponding authoritative NS RRset. This is also called the "zone apex". The "apex" is a data-theoretic description of a tree structure, and "origin" is the name of the same concept when it is implemented in zone files.

Delegation -- The process by which a separate zone is created in the name space beneath the apex of a given domain. Delegation happens when an NS RRset is added in the parent zone for the child origin, and a corresponding zone apex is created at the child origin. Delegation inherently happens at a zone cut.

In-bailiwick response -- A response in which the name server answering is authoritative for an ancestor of the owner name in the response. The term normally is used when discussing the relevancy of glue records. For example, the parent zone example.com might reply with glue records for ns.child.example.com. Because the child.example.com zone is a descendant of the example.com zone, the glue is in-bailiwick.

Out-of-bailiwick response -- A response in which the name server answering is not authoritative for an ancestor of the owner name in the response.

Authoritative data -- All of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone. (Quoted from Section 4.2.1 of

RFC 1034) It is noted that this definition might inadvertently also include any NS records that appear in the zone, even those that might not truly be authoritative because there are identical NS RRs below the zone cut. This reveals the ambiguity in the notion of authoritative data, because the parent-size NS records authoritatively indicate the delegation, even though they are not themselves authoritative data.

Root zone -- The zone whose origin is the zero-length label. Also sometimes called "the DNS root".

Empty non-terminal -- A domain name that has no RRsets, but has descendants that have RRsets. A typical example is in SRV records: in the name "_sip._tcp.example.com", it is likely that "_tcp.example.com" has no RRsets, but that "_sip._tcp.example.com" has (at least) an SRV RRset.

Delegation-centric zone -- A zone which consists mostly of delegations to child zones. This term is used in contrast to a zone which might have some delegations to child zones, but also has many data resource records for the zone itself and/or for child zones.

Wildcard -- RFC 1034 defined "wildcard", but in a way that turned out to be confusing to implementers. For an extended discussion of wildcards, including clearer definitions, see [RFC4592].

Occluded name -- The addition of a delegation point via dynamic update will render all subordinate domain names to be in a limbo, still part of the zone but not available to the lookup process. The addition of a DNAME resource record has the same impact. The subordinate names are said to be "occluded". (Quoted from [RFC5936], Section 3.5)

8. Registration Model

Registry -- The administrative operation of a zone that allows registration of names within that zone.

Registrant -- An individual or organization on whose behalf a name in a zone is registered by the registry. In many zones, the registry and the registrant may be the same entity, but in TLDs they often are not.

Registrar -- A service provider that acts as a go-between for registrants and registries. Not all registrations require a registrar, though it is common to have registrars be involved in registrations in TLDs.

EPP -- The Extensible Provisioning Protocol (EPP), which is commonly used for communication of registration information between registries and registrars. EPP is defined in [RFC5730].

9. General DNSSEC

Most DNSSEC terms are defined in [RFC4033], [RFC4034], and [RFC4035]. The terms that have caused confusion in the DNS community are highlighted here.

DNSSEC-aware and DNSSEC-unaware -- Section 2 of RFC 4033 defines many types of resolvers and validators. In specific, the terms "non-validating security-aware stub resolver", "non-validating stub resolver", "security-aware name server", "security-aware recursive name server", "security-aware resolver", "security-aware stub resolver", and "security-oblivious ’anything’" are all defined.

Signed zone -- A zone whose RRsets are signed and that contains properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records. (Quoted from RFC 4033, section 2) It has been noted in other contexts that the zone itself is not really signed, but all the relevant RRsets in the zone are signed. It should also be noted that, since the publication of [RFC6840], NSEC records are no longer required for signed zones: a signed zone might include NSEC3 records instead.

Unsigned zone -- Section 2 of RFC 4033 defines this as "a zone that is not signed". Section 2 of RFC 4035 defines this as "A zone that does not include these records [properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records] according to the rules in this section". There is an important note at the end of Section 5.2 of RFC 4035 adding an additional situation when a zone is considered unsigned: "If the resolver does not support any of the algorithms listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned."

NSEC -- The NSEC resource record lists two separate things: the next owner name (in the canonical ordering of the zone) that contains authoritative data or a delegation point NS RRset, and the set of RR types present at the NSEC RR’s owner name. (Quoted from Section 4 of 4034)

NSEC3 -- The NSEC3 resource record is quite different than the NSEC resource record. NSEC3 resource records are defined in [RFC5155].

Opt-out -- The Opt-Out Flag indicates whether this NSEC3 RR may cover unsigned delegations. (Quoted from Section 3.1.2.1 of RFC 5155)

DNSSEC Policy (DP) -- A statement that sets forth the security requirements and standards to be implemented for a DNSSEC-signed zone. (Quoted from [RFC6841], section 2)

DNSSEC Practice Statement (DPS) -- A practices disclosure document that may support and be a supplemental document to the DNSSEC Policy (if such exists), and it states how the management of a given zone implements procedures and controls at a high level. (Quoted from RFC 6841, section 2)

Key signing key (KSK) -- DNSSEC keys that only sign the apex DNSKEY RRset in a zone. (Quoted from [RFC6781], Section 3.1)

Zone signing key (ZSK) -- DNSSEC keys that can be used to sign all the RRsets in a zone that require signatures, other than the apex DNSKEY RRset. (Quoted from RFC 6781, Section 3.1)

10. DNSSEC States

A validating resolver can determine that a response is in one of four states: secure, insecure, bogus, or indeterminate. These states are defined in RFC 4033 and 4035, although the two definitions differ a bit.

Section 5 of RFC 4033 says:

A validating resolver can determine the following 4 states:

Secure: The validating resolver has a trust anchor, has a chain of trust, and is able to verify all the signatures in the response.

Insecure: The validating resolver has a trust anchor, a chain of trust, and, at some delegation point, signed proof of the non-existence of a DS record. This indicates that subsequent branches in the tree are provably insecure. A validating resolver may have a local policy to mark parts of the domain space as insecure.

Bogus: The validating resolver has a trust anchor and a secure delegation indicating that subsidiary data is signed, but the response fails to validate for some reason: missing signatures, expired signatures, signatures with unsupported algorithms, data missing that the relevant NSEC RR says should be present, and so forth.

Indeterminate: There is no trust anchor that would indicate that a specific portion of the tree is secure. This is the default operation mode.

Section 4.3 of RFC 4035 says:

A security-aware resolver must be able to distinguish between four cases:

Secure: An RRset for which the resolver is able to build a chain of signed DNSKEY and DS RRs from a trusted security anchor to the RRset. In this case, the RRset should be signed and is subject to signature validation, as described above.

Insecure: An RRset for which the resolver knows that it has no chain of signed DNSKEY and DS RRs from any trusted starting point to the RRset. This can occur when the target RRset lies in an unsigned zone or in a descendent of an unsigned zone. In this case, the RRset may or may not be signed, but the resolver will not be able to verify the signature.

Bogus: An RRset for which the resolver believes that it ought to be able to establish a chain of trust but for which it is unable to do so, either due to signatures that for some reason fail to validate or due to missing data that the relevant DNSSEC RRs indicate should be present. This case may indicate an attack but may also indicate a configuration error or some form of data corruption.

Indeterminate: An RRset for which the resolver is not able to determine whether the RRset should be signed, as the resolver is not able to obtain the necessary DNSSEC RRs. This can occur when the security-aware resolver is not able to contact security-aware name servers for the relevant zones.

11. IANA Considerations

This document has no effect on IANA registries.

12. Security Considerations

These definitions do not change any security considerations for the DNS.

13. Acknowledgements

The authors gratefully acknowledge all of the authors of DNS-related RFCs that proceed this one. Comments from Tony Finch, Stephane Bortzmeyer, Niall O’Reilly, Colm MacCarthaigh, Ray Bellis, John Kristoff, and others have helped shape this document. [[More acks will go here as people point out new terms to add and changes to the ones we have listed here.]]

14. References

14.1. Normative References

[ISO3166] International Organization for Standardization (ISO), "Country Codes - ISO 3166", February 2015, <http://www.iso.org/iso/country_codes/country_codes>.

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, November 1983.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC1206] Malkin, G. and A. Marine, "FYI on Questions and Answers: Answers to commonly asked "new Internet user" questions", RFC 1206, February 1991.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, July 2006.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.

[RFC5936] Lewis, E. and A. Hoenes, "DNS Zone Transfer Protocol (AXFR)", RFC 5936, June 2010.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, December 2012.

[RFC6840] Weiler, S. and D. Blacka, "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, February 2013.

[RFC6841] Ljunggren, F., Eklund Lowinder, AM., and T. Okubo, "A Framework for DNSSEC Policies and DNSSEC Practice Statements", RFC 6841, January 2013.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, September 2014.

14.2. Informative References

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, August 2009.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.

Authors' Addresses

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
USA

Email: paul.hoffman@vpnc.org

Andrew Sullivan
Dyn
150 Dow St, Tower 2
Manchester, NH 1604
USA

Email: asullivan@dyn.com

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Domain Name System Operations (dnsop) Working Group
Internet-Draft
Intended status: Experimental
Expires: July 11, 2016

S. Bortzmeyer
AFNIC

January 8, 2016

DNS query name minimisation to improve privacy
draft-ietf-dnsop-qname-minimisation-09

Abstract

This document describes a technique to improve DNS privacy, a technique called "QNAME minimisation", where the DNS resolver no longer sends the full original QNAME to the upstream name server.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and background

The problem statement is described in [RFC7626]. The terminology ("QNAME", "resolver", etc) is also defined in this companion document. This specific solution is not intended to fully solve the DNS privacy problem; instead, it should be viewed as one tool amongst many.

QNAME minimisation follows the principle explained in section 6.1 of [RFC6973]: the less data you send out, the fewer privacy problems you have.

Currently, when a resolver receives the query "What is the AAAA record for www.example.com?", it sends to the root (assuming a cold resolver, whose cache is empty) the very same question. Sending the full QNAME to the authoritative name server is a tradition, not a protocol requirement. This tradition comes [mockapetris-history] from a desire to optimize the number of requests, when the same name server is authoritative for many zones in a given name (something which was more common in the old days, where the same name servers served .com and the root) or when the same name server is both recursive and authoritative (something which is strongly discouraged now). Whatever the merits of this choice at this time, the DNS is quite different now.

2. QNAME minimisation

The idea is to minimise the amount of data sent from the DNS resolver to the authoritative name server. In the example in the previous section, sending "What are the NS records for .com?" would have been sufficient (since it will be the answer from the root anyway). The rest of this section describes the recommended way to do QNAME minimisation, the one which maximimes privacy benefits (other alternatives are discussed in appendixes).

A resolver which implements QNAME minimisation, and which does not have already the answer in its cache, instead of sending the full QNAME and the original QTYPE upstream, sends a request to the name server authoritative for the closest known ancestor of the original QNAME. The request is done with:

the QTYPE NS,

the QNAME which is the original QNAME, stripped to just one label more than the zone for which the server is authoritative.

For example, a resolver receives a request to resolve foo.bar.baz.example. Let’s assume it already knows that ns1.nic.example is authoritative for .example and the resolver does not know a more specific authoritative name server. It will send the query QTYPE=NS,QNAME=baz.example to ns1.nic.example.

The minimising resolver works perfectly when it knows the zone cut (zone cuts are described in section 6 of [RFC2181]). But zone cuts do not necessarily exist at every label boundary. If we take the name www.foo.bar.example, it is possible that there is a zone cut between "foo" and "bar" but not between "bar" and "example". So, assuming the resolver already knows the name servers of .example, when it receives the query "What is the AAAA record of www.foo.bar.example", it does not always know where the zone cut will be. To find it out, it will query the .example name servers for the NS records for bar.example. It will get a NODATA response, indicating there is no zone cut at that point, so it has to to query the .example name servers again with one more label, and so on. (Appendix A describes this algorithm in deeper details.)

Since the information about the zone cuts will be stored in the resolver’s cache, the performance cost is probably reasonable. Section 6 discusses this performance discrepancy further.

Note that DNSSEC-validating resolvers already have access to this information, since they have to know the zone cut (the DNSKEY record set is just below, the DS record set just above).

3. Possible issues

QNAME minimisation is legal, since the original DNS RFC do not mandate sending the full QNAME. So, in theory, it should work without any problems. However, in practice, some problems may occur (see an analysis in [huque-qnamemin] and an interesting discussion in [huque-qnamestorify]).

Some broken name servers do not react properly to qtype=NS requests. For instance, some authoritative name servers embedded in load balancers reply properly to A queries but send REFUSED to NS queries. This behaviour is a protocol violation, and there is no need to stop improving the DNS because of such behaviour. However, QNAME minimisation may still work with such domains since they are only leaf domains (no need to send them NS requests). Such setup breaks more than just QNAME minimisation. It breaks negative answers, since the servers don’t return the correct SOA, and it also breaks anything dependent upon NS and SOA records existing at the top of the zone.

Another way to deal with such incorrect name servers would be to try with QTYPE=A requests (A being chosen because it is the most common and hence a qtype which will be always accepted, while a qtype NS may ruffle the feathers of some middleboxes). Instead of querying name servers with a query "NS example.com", we could use "A _.example.com" and see if we get a referral.

A problem can also appear when a name server does not react properly to ENT (Empty Non-Terminals). If ent.example.com has no resource records but foobar.ent.example.com does, then ent.example.com is an ENT. A query, whatever the qtype, for ent.example.com must return NODATA (NOERROR / ANSWER: 0). However, some name servers incorrectly return NXDOMAIN for ENTs. If a resolver queries only foobar.ent.example.com, everything will be OK but, if it implements QNAME minimisation, it may query ent.example.com and get a NXDOMAIN. See also section 3 of [I-D.vixie-dnsext-resimprove] for the other bad consequences of this bad behaviour.

A possible solution, currently implemented in Knot, is to retry with the full query when you receive a NXDOMAIN. It works but it is not ideal for privacy.

Other practices that do not conform to the DNS protocol standards may pose a problem: there is a common DNS trick used by some Web hosters that also do DNS hosting that exploits the fact that the DNS protocol (pre-DNSSEC) allows certain serious misconfigurations, such as parent and child zones disagreeing on the location of a zone cut. Basically, they have a single zone with wildcards for each TLD like:

```
*.example.          60  IN  A   192.0.2.6
```

(They could just wildcard all of "*.", which would be sufficient. We don’t know why they don’t do it.)

This lets them have many Web hosting customers without having to configure thousands of individual zones on their nameservers. They just tell the prospective customer to point their NS records at the hoster’s nameservers, and the Web hoster doesn’t have to provision anything in order to make the customer’s domain resolve. NS queries to the hoster will therefore not give the right result, which may endanger QNAME minimisation (it will be a problem for DNSSEC, too).

4. Protocol and compatibility discussion

QNAME minimisation is compatible with the current DNS system and therefore can easily be deployed; since it is a unilateral change to the resolver, it does not change the protocol. (Because it is an unilateral change, resolver implementers may do QNAME minimisation in slightly different ways, see the appendices for examples.)

One should note that the behaviour suggested here (minimising the amount of data sent in QNAMEs from the resolver) is NOT forbidden by the [RFC1034] (section 5.3.3) or [RFC1035] (section 7.2). As said in Section 1, the current method, sending the full QNAME, is not mandated by the DNS protocol.

It may be noticed that many documents explaining the DNS and intended for a wide audience, incorrectly describe the resolution process as using QNAME minimisation, for instance by showing a request going to the root, with just the TLD in the query. As a result, these documents may confuse the privacy analysis of the users who see them.

5. Operational considerations

The administrators of the forwarders, and of the authoritative name servers, will get less data, which will reduce the utility of the statistics they can produce (such as the percentage of the various QTYPEs) [kaliski-minimum].

DNS administrators are reminded that the data on DNS requests that they store may have legal consequences, depending on your jurisdiction (check with your local lawyer).

6. Performance considerations

The main goal of QNAME minimisation is to improve privacy by sending less data. However, it may have other advantages. For instance, if a root name server receives a query from some resolver for A.example followed by B.example followed by C.example, the result will be three NXDOMAINs, since .example does not exist in the root zone. Under query name minimisation, the root name servers would hear only one question (for .example itself) to which they could answer NXDOMAIN, thus opening up a negative caching opportunity in which the full resolver could know a priori that neither B.example or C.example could exist. Thus in this common case the total number of upstream queries under QNAME minimisation would be counter-intuitively less than the number of queries under the traditional iteration (as described in the DNS standard).

QNAME minimisation may also improve look-up performance for TLD operators. For a typical TLD, delegation-only, and with delegations just under the TLD, a 2-label QNAME query is optimal for finding the delegation owner name.

QNAME minimisation can decrease performance in some cases, for instance for a deep domain name (like www.host.group.department.example.com where host.group.department.example.com is hosted on example.com’s name servers). Let’s assume a resolver which knows only the name servers of .example. Without QNAME minimisation, it would send these .example nameservers a query for www.host.group.department.example.com and immediately get a specific referral or an answer, without the need for more queries to probe for the zone cut. For such a name, a cold resolver with QNAME minimisation will, depending how QNAME minimisation is implemented, send more queries, one per label. Once the cache is warm, there will be no difference with a traditional resolver. Actual testing is described in [huque-qnamemin]. Such deep domains are specially common under ip6.arpa.

7. On the experimentation

This document has status "Experimental". Since the beginning of time (or DNS), the fully qualified host name was always sent to the authoritative name servers. There was a concern that changing this behavior may engage the Law of Unintended Consequences. Hence this status.

The idea about the experiment is to observe QNAME minimisation in action with multiple resolvers, various authoritative name servers, etc.

8. IANA Considerations

This document has no actions for IANA.

9. Security Considerations

QNAME minimisation's benefits are clear in the case where you want to decrease exposure to the authoritative name server. But minimising the amount of data sent also, in part, addresses the case of a wire sniffer as well as the case of privacy invasion by the servers. (Encryption is of course a better defense against wire sniffers but, unlike QNAME minimisation, it changes the protocol and cannot be deployed unilaterally. Also, the effect of QNAME minimisation on wire sniffers depends on whether the sniffer is, on the DNS path.)

QNAME minimisation offers zero protection against the recursive resolver, which still sees the full request coming from the stub resolver.

All the alternatives mentioned in Appendix B decrease privacy in the hope of improving performance. They must not be used if you want the maximum privacy.

10. Acknowledgments

Thanks to Olaf Kolkman for the original idea during a KLM flight from Amsterdam to Vancouver, although the concept is probably much older [1]. Thanks for Shumon Huque and Marek Vavrusa for implementation and testing. Thanks to Mark Andrews and Francis Dupont for the interesting discussions. Thanks to Brian Dickson, Warren Kumari, Evan Hunt and David Conrad for remarks and suggestions. Thanks to Mohsen Souissi for proofreading. Thanks to Tony Finch for the zone cut algorithm in Appendix A and for discussion of the algorithm. Thanks to Paul Vixie for pointing out that there are practical advantages (besides privacy) to QNAME minimisation. Thanks to Phillip Hallam-Baker for the fallback on A queries, to deal with broken servers. Thanks to Robert Edmonds for an interesting anti-pattern.

11. References

11.1. 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>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <http://www.rfc-editor.org/info/rfc7626>.

11.2. Informative References

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

[I-D.wkumari-dnsop-hammer]
Kumari, W., Arends, R., Woolf, S., and D. Migault, "Highly Automated Method for Maintaining Expiring Records", draft-wkumari-dnsop-hammer-01 (work in progress), July 2014.

[I-D.vixie-dnsext-resimprove]
Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsext-resimprove-00 (work in progress), June 2010.

[mockapetris-history]
Mockapetris, P., "Private discussion", January 2015.

[kaliski-minimum]
Kaliski, B., "Minimum Disclosure: What Information Does a Name Server Need to Do Its Job?", March 2015, <http://blogs.verisigninc.com/blog/entry/minimum_disclosure_what_information_does>.

[huque-qnamemin]
Huque, S., "Query name minimization and authoritative server behavior", May 2015, <https://indico.dns-oarc.net/event/21/contribution/9>.

[huque-qnamestorify]
Huque, S., "Qname Minimization @ DNS-OARC", May 2015, <https://storify.com/shuque/qname-minimization-dns-oarc>.

11.3. URIs

[1] https://lists.dns-oarc.net/pipermail/dns-operations/2010-February/005003.html

Appendix A. An algorithm to perform QNAME minimisation

This algorithm performs name resolution with QNAME minimisation in presence of not-yet-known zone cuts.

Although a validating resolver already has the logic to find the zone cut, other resolvers may be interested by this algorithm to follow in order to locate the cuts. This is just a possible help for implementors, it is not intended to be normative:

(0) If the query can be answered from the cache, do so, otherwise iterate as follows:

(1) Find closest enclosing NS RRset in your cache. The owner of this NS RRset will be a suffix of the QNAME - the longest suffix of any NS RRset in the cache. Call this ANCESTOR.

(2) Initialize CHILD to the same as ANCESTOR.

(3) If CHILD is the same as the QNAME, resolve the original query using ANCESTOR’s name servers, and finish.

(4) Otherwise, add a label from the QNAME to the start of CHILD.

(5) If you have a negative cache entry for the NS RRset at CHILD, go back to step 3.

(6) Query for CHILD IN NS using ANCESTOR’s name servers. The response can be:

(6a) A referral. Cache the NS RRset from the authority section and go back to step 1.

(6b) An authoritative answer. Cache the NS RRset from the answer section and go back to step 1.

(6c) An NXDOMAIN answer. Return an NXDOMAIN answer in response to the original query and stop.

(6d) A NOERROR/NODATA answer. Cache this negative answer and go back to step 3.

Appendix B. Alternatives

Remember that QNAME minimisation is unilateral so a resolver is not forced to implement it exactly as described here.

There are several ways to perform QNAME minimisation. The one in Section 2 is the suggested one. It can be called the aggressive algorithm, since the resolver only sends NS queries as long as it does not know the zone cuts. This is the safest, from a privacy point of view. Another possible algorithm, not fully studied at this time, could be to "piggyback" on the traditional resolution code. At startup, it sends traditional full QNAMEs and learns the zone cuts from the referrals received, then switches to NS queries asking only for the minimum domain name. This leaks more data but could require fewer changes in the existing resolver codebase.

In the above specification, the original QTYPE is replaced by NS (or may be A, if too many servers react incorrectly to NS requests), which is the best approach to preserve privacy. But this erases information about the relative use of the various QTYPEs, which may be interesting for researchers (for instance if they try to follow IPv6 deployment by counting the percentage of AAAA vs. A queries). A variant of QNAME minimisation would be to keep the original QTYPE.

Another useful optimisation may be, in the spirit of the HAMMER idea [I-D.wkumari-dnsop-hammer] to probe in advance for the introduction of zone cuts where none previously existed (i.e. confirm their continued absence, or discover them.)

To address the "number of queries" issue, described in Section 6, a possible solution is to always use the traditional algorithm when the cache is cold and then to move to QNAME minimisation (precisely defining what is "hot" or "cold" is left to the implementer). This will decrease the privacy but will guarantee no degradation of performance.

Author’s Address

Stephane Bortzmeyer
AFNIC
1, rue Stephenson
Montigny-le-Bretonneux 78180
France

Phone: +33 1 39 30 83 46
Email: bortzmeyer+ietf@nic.fr
URI: http://www.afnic.fr/

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 3, 2016

W. Kumari
Google
P. Hoffman
ICANN

October 1, 2015

Decreasing Access Time to Root Servers by Running One on Loopback
draft-ietf-dnsop-root-loopback-05

Abstract

Some DNS recursive resolvers have longer-than-desired round trip times to the closest DNS root server. Some DNS recursive resolver operators want to prevent snooping of requests sent to DNS root servers by third parties. Such resolvers can greatly decrease the round trip time and prevent observation of requests by running a copy of the full root zone on a loopback address (such as 127.0.0.1). This document shows how to start and maintain such a copy of the root zone that does not pose a threat to other users of the DNS, at the cost of adding some operational fragility for the operator.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DNS recursive resolvers have to provide answers to all queries from their customers, even those which are for domain names that do not exist. For each queried name that has a top level domain (TLD) that is not in the recursive resolver's cache, the resolver must send a query to a root server to get the information for that TLD, or to find out that the TLD does not exist. Typically, the vast majority of queries going to the root are for names that do not exist in the root zone, and the negative answers are cached for a much shorter period of time. A slow path between the recursive resolver and the closest root server has a negative effect on the resolver's customers.

Recursive resolvers currently send queries for all TLDs that are not in their caches to root servers, even though most of those queries get answers that are referrals to other servers. Malicious third parties might be able to observe that traffic on the network between the recursive resolver and one or more of the DNS roots.

This document describes a method for the operator of a recursive resolver to greatly speed these queries and to hide them from

outsiders. The basic idea is to create an up-to-date root zone server on a loopback address on the same host as the recursive server, and use that server when the recursive resolver looks up root information. The recursive resolver validates all responses from the root server on the loopback address, just as it would all responses from a remote root server.

The primary goals of this design is to provide faster negative responses to stub resolver queries that contain junk queries, and to prevent queries and responses from being visible on the network. This design will probably have little effect on getting faster positive responses to stub resolver for good queries on TLDs, because the data for those zones is usually long-lived and already in the cache of the recursive resolver; thus, getting faster positive responses is a non-goal of this design.

This design explicitly only allows the new root zone server to be run on a loopback address, in order to prevent the server from serving authoritative answers to any system other than the recursive resolver.

It is important to note that this design is being described here is not considered a "best practice". In fact, many people feel that it is an excessively risky practice because it introduces a new operational piece to local DNS operations where there was not one before. The advantages listed above do not come free: if this new system does not work correctly, users can get bad data, or the entire recursive resolution system might fail in ways that are hard to diagnose.

This design requires the addition of authoritative name server software running on the same machine as the recursive resolver. Thus, recursive resolver software such as BIND will not need to add much new functionality, but recursive resolver software such as Unbound will need to be able to talk to an authoritative server (such as NSD) running on the same host.

Because of the significant operational risks described in this document, distributions of recursive DNS servers MUST NOT include configuration for the design described here. It is acceptable to point to this document, but not to indicate that this configuration is something that should be considered without reading the entire document.

A different approach to solving the problems discussed in this document is described in [AggressiveNSEC].

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Requirements

In order to implement the mechanism described in this document:

- o The system MUST be able to validate a zone with DNSSEC [RFC4033].
- o The system MUST have an up-to-date copy of the DNS root key.
- o The system MUST be able to retrieve a copy of the entire root zone (including all DNSSEC-related records).
- o The system MUST be able to run an authoritative server on one of the IPv4 loopback addresses (that is, an address in the range 127/8 for IPv4 or ::1 in IPv6).

A corollary of the above list is that authoritative data in the root zone used on the local authoritative server MUST be identical to the same data in the root zone for the DNS. It is possible to change the unsigned data (the glue records) in the copy of the root zone, but such changes could cause problems for the recursive server that accesses the local root zone, and therefore any changes to the glue records SHOULD NOT be made.

3. Operation of the Root Zone on the Loopback Address

The operation of an authoritative server for the root in the system described here can be done separately from the operation of the recursive resolver.

The steps to set up the root zone are:

1. Retrieve a copy of the root zone. (See Appendix A for some current locations of sources.)
2. Start the authoritative server with the root zone on a loopback address that is not in use. For IPv4, this would typically be 127.0.0.1, but if that address is in use, any address in 127/8 is acceptable. For IPv6, this would be ::1.

The contents of the root zone MUST be refreshed using the timers from the SOA record in root zone, as described in [RFC1035]. This inherently means that the conents of the local root zone will likely

be a little behind those of the global root servers because those servers are updated triggered by NOTIFY messages. If the contents of the zone cannot be refreshed before the expire time, the server MUST return a SERVFAIL error response for all queries until the zone can be successfully be set up again.

In the event that refreshing the contents of the root zone fails, the results can be disastrous. For example, sometimes all the NS records for a TLD are changed in a short period of time; if the local root zone refreshing is broken during that time, the recursive resolver will have bad data for the entire TLD zone.

An administrator using the procedure in this document SHOULD have an automated method to check that the contents of the local root zone are being refreshed. One way to do this is to have a separate process that periodically checks the SOA of the root zone from the local root zone and makes sure that they are changing. At the time that this document is published, the SOA for the root zone is the digital representation of the current date with a two-digit counter appended, and the SOA is changed every day even if the contents of the root zone are unchanged. For example, the SOA of the root zone on January 2, 2015 was 2015010201. A process can use this fact to create a check for the contents of the local root zone (using a program not specified in this document).

4. Using the Root Zone Server on the Loopback Address

A recursive resolver that wants to use a root zone server operating as described in Section 3 simply specifies the local address as the place to look when it is looking for information from the root. All responses from the root server must be validated using DNSSEC.

Note that using this configuration will cause the recursive resolver to fail if the local root zone server fails. See Appendix B for more discussion of this for specific software.

To test the proper operation of the recursive resolver with the local root server, use a DNS client to send a query for the SOA of the root to the recursive server. Make sure the response that comes back has the AA bit in the message header set to 0.

5. IANA Considerations

This document requires no action from the IANA.

6. Security Considerations

A system that does not follow the DNSSEC-related requirements given in Section 2 can be fooled into giving bad responses in the same way as any recursive resolver that does not do DNSSEC validation on responses from a remote root server. Anyone deploying the method described in this document should be familiar with the operational benefits and costs of deploying DNSSEC [RFC4033].

As stated in Section 1, this design explicitly only allows the new root zone server to be run on a loopback address, in order to prevent the server from serving authoritative answers to any system other than the recursive resolver. This has the security property of limiting damage to any other system that might try to rely on the copy of the root in case that copy becomes altered.

7. Acknowledgements

The editors fully acknowledge that this is not a new concept, and that we have chatted with many people about this. In fact, this concept may already have been implemented without the knowledge of the authors. For example, Bill Manning described a similar solution but to a very different problem (intermittent connectivity, instead of constant but slow connectivity) in his doctoral dissertation in 2013 [Manning2013].

Evan Hunt contributed greatly to the logic in the requirements. Other significant contributors include Wouter Wijngaards, Tony Hain, Doug Barton, Greg Lindsay, and Akira Kato. The authors also received many off-line comments about making the document clear that this was just a description of a way to operate a root zone on localhost, and not a recommendation to do so.

8. References

8.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

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

8.2. Informative References

[AggressiveNSEC]
Fujiwara, K. and A. Kato, "Aggressive use of NSEC/NSEC3", draft-fujiwara-dnsop-nsec-aggressiveuse-00 (work in progress), 2015.

[Manning2013]
Maning, W., "Client Based Naming", 2013, <http://www.sfc.wide.ad.jp/dissertation/bill_e.html>.

Appendix A. Current Sources of the Root Zone

The root zone can be retrieved from anywhere as long as it comes with all the DNSSEC records needed for validation. Currently, one can get the root zone from ICANN by zone transfer (AXFR) over TCP from DNS servers at xfr.lax.dns.icann.org and xfr.cjr.dns.icann.org.

Currently, the root can also be retrieved by AXFR over TCP from the following root server operators:

- b.root-servers.net
- c.root-servers.net
- f.root-servers.net
- g.root-servers.net
- k.root-servers.net

It is crucial to note that none of the above services are guaranteed to be available. It is possible that ICANN or some of the root server operators will turn off the AXFR capability on the servers listed above. Using AXFR over TCP to addresses that are likely to be anycast (as the the ones above are) may conceivably have transfer problems due to anycast, but current practice shows that to be unlikely.

To repeat the requirement from earlier in this document: if the contents of the zone cannot be refreshed before the expire time, the server MUST return a SERVFAIL error response for all queries until the zone can be successfully be set up again.

Appendix B. Example Configurations of Common Implementations

This section shows fragments of configurations for some popular recursive server software that is believed to correctly implement the requirements given in this document.

The IPv4 and IPv6 addresses in this section were checked recently by testing for AXFR over TCP from each address for the known single-letter names in the root-servers.net zone.

The examples here use a loopback address of 127.12.12.12, but typical installations will use 127.0.0.1. The different address is used in order to emphasize that the root server does not need to be on the device at "localhost".

B.1. Example Configuration: BIND 9.9

BIND acts both as a recursive resolver and an authoritative server. Because of this, there is "fate sharing" between the two servers in the following configuration. That is, if the root server dies, it is likely that all of BIND is dead.

Using this configuration, queries for information in the root zone are returned with the AA bit not set.

When slaving a zone, BIND will treat zone data differently if it is slaved into a separate view (or a separate instance of the software) versus slaving the zone into the same view or instance that is also performing the recursion.

Validation: When using separate views or separate instances, the DS records in the slaved zone will be validated as the zone data is accessed by the recursive server. When using the same view, this validation does not occur for the slaved zone.

Caching: When using separate views or instances, the recursive server will cache all of the queries for the slaved zone, just as it would using the traditional root hints method. Thus, as the zone in the other view or instance is refreshed or updated, changed information will not appear in the recursive server until the TTL of the old record times out. Currently the TTL for DS and delegation NS records is two days. When using the same view, all zone data in the recursive server will be updated as soon as it receives its copy of the zone.

```
view root {
    match-destinations { 127.12.12.12; };
    zone "." {
        type slave;
        file "rootzone.db";
        notify no;
        masters {
            192.228.79.201; # b.root-servers.net
            192.33.4.12;    # c.root-servers.net
            192.5.5.241;    # f.root-servers.net
            192.112.36.4;   # g.root-servers.net
            193.0.14.129;   # k.root-servers.net
            192.0.47.132;   # xfr.cjr.dns.icann.org
            192.0.32.132;   # xfr.lax.dns.icann.org
            2001:500:84::b; # b.root-servers.net
            2001:500:2f::f; # f.root-servers.net
            2001:7fd::1;    # k.root-servers.net
            2620:0:2830:202::132;  # xfr.cjr.dns.icann.org
            2620:0:2d0:202::132;  # xfr.lax.dns.icann.org
        };
    };
};

view recursive {
    dnssec-validation auto;
    allow-recursion { any; };
    recursion yes;
    zone "." {
        type static-stub;
        server-addresses { 127.12.12.12; };
    };
};
```

B.2. Example Configuration: Unbound 1.4 and NSD 4

Unbound and NSD are separate software packages. Because of this, there is no "fate sharing" between the two servers in the following configurations. That is, if the root server instance (NSD) dies, the recursive resolver instance (Unbound) will probably keep running, but will not be able to resolve any queries for the root zone. Therefore, the administrator of this configuration might want to carefully monitor the NSD instance and restart it immediately if it dies.

Using this configuration, queries for information in the root zone are returned with the AA bit not set.

```
# Configuration for Unbound
server:
    do-not-query-localhost: no
stub-zone:
    name: "."
    stub-prime: no
    stub-addr: 127.12.12.12

# Configuration for NSD
server:
    ip-address: 127.12.12.12
zone:
    name: "."
    request-xfr: 192.228.79.201 NOKEY # b.root-servers.net
    request-xfr: 192.33.4.12 NOKEY    # c.root-servers.net
    request-xfr: 192.5.5.241 NOKEY    # f.root-servers.net
    request-xfr: 192.112.36.4 NOKEY   # g.root-servers.net
    request-xfr: 193.0.14.129 NOKEY   # k.root-servers.net
    request-xfr: 192.0.47.132 NOKEY   # xfr.cjr.dns.icann.org
    request-xfr: 192.0.32.132 NOKEY   # xfr.lax.dns.icann.org
    request-xfr: 2001:500:84::b NOKEY # b.root-servers.net
    request-xfr: 2001:500:2f::f NOKEY # f.root-servers.net
    request-xfr: 2001:7fd::1 NOKEY    # k.root-servers.net
    request-xfr: 2620:0:2830:202::132 NOKEY  # xfr.cjr.dns.icann.org
    request-xfr: 2620:0:2d0:202::132 NOKEY  # xfr.lax.dns.icann.org
```

B.3. Example Configuration: Microsoft Windows Server 2012

Windows Server 2012 contains a DNS server in the "DNS Manager" component. When activated, that component acts as a recursive server. DNS Manager can also act as an authoritative server.

Using this configuration, queries for information in the root zone are returned with the AA bit set.

The steps to configure DNS Manager to implement the requirements in this document are:

1. Launch the DNS Manager GUI. This can be done from the command line ("dnsmgmt.msc") or from the Service Manager (the "DNS" command in the "Tools" menu).

2. In the hierarchy under the server on which the service is running, right-click on the "Forward Lookup Zones", and select "New Zone". This brings up a succession of dialog boxes.

3. In the "Zone Type" dialog box, select "Secondary zone".

4. In the "Zone Name" dialog box, enter ".".

5. In the "Master DNS Servers" dialog box, enter "b.root-servers.net". The system validates that it can do a zone transfer from that server. (After this configuration is completed, DNS Manager will attempt to transfer from all of the root zone servers.)

6. In the "Completing the New Zone Wizard" dialog box, click "Finish".

7. Verify that the DNS Manager is acting as a recursive resolver. Right-click on the server name in the hierarch, choosing the "Advanced" tab in the dialog box. See that "Disable recursion (also disables forwarders)" is not selected, and that "Enable DNSSEC validation for remote responses" is selected.

Authors' Addresses

Warren Kumari
Google

Email: Warren@kumari.net

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2018

W. Mekking
Oracle Dyn
O. Gudmundsson
CloudFlare

March 28, 2018

Minimal Incremental Zone Transfer in DNS
draft-mekking-mixfr-02

Abstract

This document proposes extensions to the DNS protocol to provide an incremental zone transfer (IXFR) mechanism with dynamic update (UPDATE) capabilities, to keep IXFRs that deal with DNSSEC small.

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 September 29, 2018.

Table of Contents

1. Introduction

Incremental zone transfer (IXFR, [RFC1995]) was introduced to efficiently transfer changed portions of a zone. However, when a zone is signed with DNSSEC [RFC4033], [RFC4034], [RFC4035], the transfer can still become very large. For example, when many resource record sets (RRsets) need to be re-signed, or when the NSEC3 [RFC5155] salt is changed, an IXFR may become larger than a full zone transfer (AXFR, [RFC5936]). This is because the IXFR includes complete copies of both the deleted and replacement RRSIG records.

To keep the deltas small in zone transfers, we need to have a richer change syntax, for example like in Dynamic Update (DNS UPDATE, [RFC2136]. This document introduces a new query type MIXFR (minimal incremental zone transfer) that is able to express this richer syntax. The goal of this proposal is to allow small changes to be communicated over UDP, and remove as much redundant information from the zone transfer as possible.

An earlier proposal to keep the zone transfers small is IXFR-ONLY [IXFR-ONLY], by giving the client an oppurtunity to signal the server

that it prefers an error above a fall back to an AXFR in case the server is not able to send an IXFR. However IXFR-ONLY did not reduce the size of an IXFR.

2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Syntax

The syntax for MIXFR is a superset of IXFR. The richer syntax of MIXFR allows to add or delete multiple records with one resource record (RR). MIXFR is DNSSEC aware thus if there is a change to RRset it knows to delete the covering RRISG(s), this saves the transmission of old RRsigs.

3.1. Implicit RRSIG deletion

When an RRset is modified, the MIXFR client MUST also remove all existing RRSIG records on that RRset. This is valid for all RRtypes except RRSIG itself.

3.2. Add an RR

This works the same as with IXFR, with implicit RRSIG delete logic added.

3.3. Delete an RR

This works the same as with IXFR, with implicit RRSIG delete logic added.

3.4. Delete an RRset

Similar to DNS UPDATE. To delete an RRset, the MIXFR deletion list includes an RR whose NAME and TYPE are those of the RRset to be deleted. CLASS must be specified as ANY. RDLENGTH must be zero (0) and RDATA must therefore be empty. This also deletes the covering RRSIGs.

Note that a record with its CLASS set to ANY does _not_ mean to delete (or change) the record in all available classes: zone transfers are encapsulated in SOA records that determine the zone name and class (see Figure ()[#fig:a-MIXFR-response]). Only changes in the zone matching that name and class will be made.

3.5. Delete All RRsets on a Name

Similar to DNS UPDATE. To delete all RRSets at a name, the MIXFR deletion list includes an RR at that NAME, whose TYPE must be specified as ANY and CLASS must be specified as ANY. RDLENGTH must be zero (0) and RDATA must therefore be empty.

3.6. Replace an RRset

The MIXFR addition list includes an RR whose NAME and TYPE are those of the RRset to be replaced. CLASS must be specified as ANY. RDLENGTH must be non-zero and the RDATA is that of the first replacement record.

If an RRset is to be replaced with multiple records, the second and subsequent records MUST use the syntax for adding an RR.

The same syntax is used to delete an RRset and to replace an RRset with an RR whose RDLENGTH is zero. This is not ambiguous because the former appears in the deletion list (before the new SOA RR) and the latter appears in the addition list (after the new SOA RR).

4. Protocol Description

4.1. Client side

The client can send a MIXFR request. Just like with IXFR, it places a SOA RR in the authority section to signal the version of the zone it holds now. If the client does not want the server to fall back to AXFR, it MAY add another SOA RR in the additional section. This achieves MIXFR-only behavior, similar to IXFR-ONLY [IXFR-ONLY]. For example:

```
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 1337
;; flags: qr ; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; QUESTION SECTION:
;; example.    IN    MIXFR

;; AUTHORITY SECTION:
example.       IN    SOA    serial=1

;; ADDITIONAL SECTION:
example.       IN    SOA    serial=1
```

Figure 1: A MIXFR request for the "example." zone.

[MM] Adding a whole record is quite some overhead in bits while we only signal one bit of information: to fall back or not to fall back.

[OG] Can we use a bit from header or OPT record? Or can we just use "Class | 0x8000" to signal that?

4.2. Server side

A server receiving a minimal incremental zone transfer (MIXFR) request will reply with a MIXFR. A MIXFR looks exactly like an IXFR, except there may be zero or more of the new introduced syntax RRs that can add or delete more records. For the zone "example.", the following zone transfer can be sent that will replace all signatures in the zone with new signatures for the names "example.", "a.example.", "b.example." and "c.example.":

```
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 1337
;; flags: qr ; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;; example.    IN    MIXFR

;; ANSWER SECTION:
example.       IN    SOA    serial=3
example.       IN    SOA    serial=1
example.       ANY   RRSIG
example.       IN    SOA    serial=3
example.       IN    RRSIG  rdata
a.example.     IN    RRSIG  rdata
b.example.     IN    RRSIG  rdata
c.example.     IN    RRSIG  rdata
example.       IN    SOA    serial=3
```

Figure 2: A MIXFR response for the "example." zone.

The server MAY reply with an IXFR or AXFR instead. If the server does not implement MIXFR it MUST return a response with NOTIMPL rcode. The client MUST fallback to request IXFR or AXFR.

4.3. Future zone transfer improvements

In many cases DNS servers have many zones in common, and there are many changes in the zones each hour, in this case having a long lived TCP connection or an out-of-band protocol where the primary server can push changes to the secondary.

The size of the zone transfer can be reduced even more if the syntax on the wire is changed, i.e. the RR wire format is abandoned. A different grammar may add operators, remove duplicate RRset owner names, and use standard compression algorithms.

These kind of improvements will require more drastic changes, and may be covered in a separate, future document.

5. IANA Considerations

IANA is requested to assign the OPCODE value [TBD] (decimal) for MIXFR, in sub-registry "DNS OpCodes" of registry "Domain Name System (DNS) Parameters".

6. Security Considerations

This document does not introduce additional security considerations. Or does it?

Should we explain what the security implications are, because descriptions from old RFC’s are not good enough?

Any MIXFR transactions should use secure channels such as IPSEC or SSH tunnel, and use TSIG for authentication.

7. Acknowledgements

Johan Ihren, Tony Finch, Bob Harold.

8. References

8.1. Informative References

[IXFR-ONLY] Sury, O. and S. Kerr, "IXFR-ONLY to Prevent IXFR Fallback to AXFR", February 2010, <https://tools.ietf.org/html/draft-kerr-ixfr-only-01>.

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

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[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>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

8.2. Normative References

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <https://www.rfc-editor.org/info/rfc1995>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

Appendix A. Changelog

A.1. Version 02

- Removed 'Delete All RRsets of a Type' because it had the same syntax as 'Delete an RRset' [Olafur].
- Clarify ANY CLASS [#5, Bob Harold].
- Sleep for 3 years.
- Remove IXFR Gone Wild section.

A.2. Version 01

- Split document in trivial and 'more wild' ideas.

A.3. Version 00

- Initial version

Authors' Addresses

W. (Matthijs) Mekking
Oracle Dyn
Hertogswetering 163-167
Utrecht 3543 AS Utrecht
NL

EMail: matthijs.mekking@oracle.com
URI: https://www.dyn.com

Olafur Gudmundsson
CloudFlare
San Francisco, CA 94107
USA

EMail: olafur@cloudflare.com

DNSOP
Internet-Draft
Updates: 1035 (if approved)
Intended status: Standards Track
Expires: September 10, 2015

O. Gudmundsson
M. Majkowski
CloudFlare Inc.
J. Abley
Dyn, Inc.

March 9, 2015

DNS Meta-Queries resticted.
draft-ogud-dnsop-acl-metaqueries-00

Abstract

Some DNS types have special meaning and are classified as meta queries, this includes ANY, AXFR, IXFR. These queries frequently return larger answers than queries for other types.

This document defines a standard way for Authoritative-Only servers how to refuse to serve these and other similar queries, with the expectation that resolvers honor that, by not asking followup queries.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DNS Specification [RFC1035] meta queries where defined for use either zone maintainance AXFR, full zone transfer, IXFR [RFC1995], incremental zone transfer. For security reasons Authoritative name servers frequently only respond to these queries if a TSIG [RFC2845] key is presented or the query comes from an approved address.

The ANY meta query was defined for debugging purposes mainly against resolvers. There have been widespread misunderstanding as to what the query is supposed to do and when it is approriate. The query is intented for testing what records for a particular name a resolver has in its cache. There are security implications related to information leaks and use in DoS attacks that strongly argue for restricting its use like the other Meta Queries.

RRSIG [RFC4034] type used in a query can also return large answers as the server attempts to put all RRSIG records at that one name into one answer. This type was envisioned as deployment tool for validators to overcome DNSSEC ignorant resolvers and/or servers. For all practical purposes this is never needed.

Queries yielding large answers are known to be widely abused by attackers carrying out reflection attacks, since they provide a convenient way to elicit large responses from small queries, and hence exhibit significant amplification potential. A similar reaction to an operational security problem can be observed in the advice contained within [RFC5358].

The data model used by some authoritative-only DNS server implementations does not align easily with the zone structure described in [RFC1035], and responding accurately to meta queries involves significant processing overhead. The ability to refuse meta queries can simplify the implementation with corresponding benefits to performance and code correctness.

Recursive Resovlers frequently treat REFUSED as a temporary denial. In the case of policy statement that certain queries will not answered, having a more explicit statment is beneficial. There are two chocies as how more permanent semantics can be expressed, reusing an exisiting RCODE or define a new one. This docuemnt proposes reusing the NOTIMP rcode. This feels like the right choice as as far as the querier is concerned it makes no difference if the meta type is implemented or the authoritative server has no interest in providing that service to the client. There are other options like defining new RCODE or place stronger semantics on REFUSED.

Various DNS operators have chosen to refuse various meta queries including QTYPE=ANY in the past, using a variety of approaches, including rate-limiting of queries and responses, returning TC=1 on qeries received via UDP transport and silently dropping queries before they reach the DNS server. Consistency in approach would provide a more predictable outcome for DNS resolvers and clients.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Protocol Changes

DNS processing entities SHOULD support authenticated meta queries, and process them when appropriate as defined by policy. By default the implemenations SHOULD be restrict it to localhost via ACL. For all rejected meta-queries the behavior specified below SHOULD be used. The types where this behavior is appropriate includes ANY, AXFR, IXFR, RRSIG.

An authoritative-only DNS server MAY reject meta queries received by returning RCODE=4 (NOTIMP).

An iterative resolver MUST NOT forward a meta-query when the query arrives with RD=0, even when it has no types for that name. An iterative resolver SHOULD ignore RD=1 on a meta query, i.e. it SHOULD NOT forward them upstream.

An iterative resolver that sends a query to an authoritative DNS server and receives a response with RCODE=4 SHOULD remember that upstream server’s behaviour, for that qclass, qname, qtype combination. It SHOULD suppress any subsequent qeries for that qclass, qname, qtype to that server for at least one day (??? better value needed).

4. IANA Considerations

No actions are requested of the IANA.

5. Security Considerations

In the original Internet where everyone behaved nicely had different secrity and operating model than todays Internet. This document is defining how DNS servers can express that they will never answer a particular query from a given address.

RCODE=REFUSED is frequently treated as temporary thus resolver may repeat queries in the hope of getting an answer.

An on-path attacker[RFC3833] can forge these answers easily, but as that document explains the attacker can anyway inject any lies it wants to.

6. Implementation Experience

TBD

7. Acknowledgements

Editors want to thank following people, in random order, for useful feedback: Paul Vixie, Tony Finch, Ralph Weber, Mark Andrews, Stephane ortzmeyre, Filippo Valsodra, Edward Lewis, and we forgot someone.

8. References

8.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, August 1996.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, October 2008.

Appendix A. Document history

This section (and sub-sections) should be removed before publication.

A.1. Venue

An appropriate venue to discuss this draft is the dnsop working group mailing list.

A.2. Abridged Revision History

A.2.1. draft-ogud-dnsop-any-notimp-00

Initial draft.

A.2.2. draft-ogud-dnsop-acl-metaqueries-00

Wordsmithing; add jabley as co-author; normalise normative language in protocol changes section.

Based on feeback from dnsop mailing list, we Expaned the scope of the document to cover "META" types in general, and express that RCODE=NOTIMP should be cached by resolvers. Changed language so it is more neutral to as what path this work takes.

Authors' Addresses

Olafur Gudmundsson
CloudFlare Inc.
San Francisco, CA 94107
USA

Email: olafur@cloudflare.com

Marek Majkowski
CloudFlare Inc.
London
UK

Email: marek@cloudflare.com

Joe Abley
Dyn, Inc.
103-186 Albert Street
London, ON N6A 1M1
Canada

Email: jabley@dyn.com

DNSOP Working Group
Internet Draft
Intended status: Informational
Expires: August 2015

Wei Wang
Zhiwei Yan
NANEL

February 24, 2015

A Survey of the DNS cache service in China
draft-wang-dnsop-cachesurvey-00.txt

Abstract

DNS cache directly serves the DNS queries from stub resolvers as the data source in the specified network area. For the present, however, operators manage and run the cache service in a diversified manner. This arouses the main motivation of this survey report. Instead of regulating or specifying the operation of the DNS cache service, our aim is to investigate the situation of the DNS cache service (at least in mainland China) and propose the future operation recommendations with solid practical foundation.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of this Memo

Table of Contents

1. Survey respondents

This survey covers three main Internet service providers (ISPs) in China and the top three recursive service providers in China, as following:

1) China Telecom Co.Ltd.

2) China United Network Communications Group Co.Ltd.

3) China Mobile Communications Co.,Ltd.

4) Qihoo 360 Technology Co. Ltd.

5) Alibaba Group Holding Ltd.

6) Tencent Holdings Ltd.

2. Survey results

So as to present the survey results clearly and concisely, we select only the key results and have them listed with analytical logics.

2.1. Overview

In order to make this survey rational, six most representative survey respondents are selected. Half of them are typical ISPs and the others are typical public recursive service providers in China. All the six survey respondents deploy recursive service quite widely with stable service scale.

(In consider of the business secret protection, the geographical coverage, amount of the clients and service scale of the survey respondents are not given here because it is inappropriate to show them together.)

2.2. Architecture improvement of recursive service

To meet the respective demands of business operation and IT operation, recursive service operators simultaneously take the same architecture model, transformingthe classical textbookish recursive server into a composite architecture consisted of three independent servers: online cache, recursive server and offline (or backup) server. We denote this kind of recursive service architecture as "Big recursive service" in view of its large scale and serious influence, as shown in Figure 1.

```
                        +------------------------+
                        |                        |
                        |      +-+-+-+-+         |
                        |      |Backup |         |
                        |      |server |         |
                        |      +-+-+-+-+         |
                        |                        |
          +-+-+-+-+-+   | +-+-+-+-+  +-+-+-+-+-+ |  +-+-+-+-+-+-+-+-+
          |Stub     |   | |Online |  |Recursive| |  |Authoritative|
          |resolver |-----|cache  |  |server   |----|server       |
          +-+-+-+-+-+   | +-+-+-+-+  +-+-+-+-+-+ |  +-+-+-+-+-+-+-+-+
                        |                        |
                        |  Big recursive service |
                        +------------------------+
```

Figure 1. Big recursive service model

Specifically, the online cache serves the stub resolvers directly, and the backup server is mainly used in the emergency case as a

backup data source, while the recursive server fetches DNS data from the authoritative servers.

2.3. Local cache service

All the six survey respondents deploy the local cache service. Due to different business requirements, they all cache the TOP-N domain names, while three of them cache the root and TLD zone files as well.

(We here use the term "local" to manifest the administration boundary of the service, such as province region of an ISP, covering area of DNS end users and etc.)

2.3.1. Root zone file cache

For the three ISPs, they all cache the root zone file.

The actual requirement to cache the root zone file is for the emergency response and it is not used as online service. For each ISP, the root zone file cache is deployed in one server instance in a shared manner (in province level) to cover all the recursive servers in its related autonomous area.

The data is updated once per day from open data source, but the integrity and correctness of the downloaded data are not verified (for example with DNSSEC).

2.3.2. TLD zone file cache

For the three ISPs, all of them cache some TLD zone files. The actual requirement and deployment model of the TLD zone file cache is the same as the case of root zone file cache.

The data is updated once per day from open data source but the integrity and correctness of the downloaded data are not verified (for example with DNSSEC).

2.3.3. TOP-N domain names cache

All the six survey respondents cache the resource records of TOP-N domain names. The selected TOP-N domain names are different between different survey respondents based on respective online service log and scale. But the scales of cached domain names can varied from 1 million to 100 million regarding to the amount of end user and the business policy of operators.

The cached data is directly used for responding the requests from the stub resolvers in order to satisfy the stub resolvers most efficiently. Besides, the cached data is maintained in an active manner with some respondents, for example, some recursive

servers anticipate the expiration of the cached data and fetch it without receiving the actual request from client.

3. Analysis

In the following, the positive and negative impacts of the "Big recursive service" on the DNS ecosystem are analyzed:

1) Online cache

a) Positive points: The online cache of the six survey respondents is in large amount, almost above million levels. In this way, stub resolvers can be served efficiently and it reduces the impacts of attacks towards the recursive server.

b) Negative points: It will break the balance of the classical DNS model as the query amount of authoritative server is inversely proportional to the cache scale. The amount of queries will decrease with the enlargement of online cache. In an extreme case, the authoritative server could recognize only one request from China during valid TTL period if only one single online cache covers all DNS requests in China.

2) Backup server

a) Positive points: The backup server is maintained in order to recover the DNS resolution service in the emergency case.There are two types of data in the backup server: a) zone files (including the root and TLDs); b) snapshot of the online service.

b) Negative points: Currently, backup server can be activated by the operator without notifying the related authoritative server. It means that the authoritative server will be completely replaced by backup server in emergency area, and queries from that area will drop steeply even till to zero.

3) Recursive server

a) Positive points: The load of the recursive server will be decreased significantly. And it only focuses on the communication with authoritative server. In this way, the operation and failure risk will reduce.

b) Negative points: Due to the above mentioned cache functions, recursive server has degenerated as the "weak" tool, which only fetches and refreshes the authoritative data in the cache or helps scheduling some sophisticated applications like CDN service (e.g., to schedule the client to the suitable server instance according to the geographical location of the client).

In this way, requests sent from recursive server to authoritative server may not be actually triggered by stub resolvers, or if they are wholly simulated, it will result in the distortion of the query behavior at authoritative server, and the judgment of administrator will be affected correspondingly.

Survey contributors

The following individuals served as experts and representatives of the survey respondents during the completion of this survey report. The contributions from their respective experience as a stakeholder, a corporate manager or technical expert had bestowed essential guidance to the analysis and conclusions presented herein. Contributors may not agree with all the observations statedin the document, but all agree that it presents an important reference for succeeding works. In addition to those listed below, there were an equal number of contributors with equal stature whose names are not included for various reasons.

Ziqian Liu
China Telecom Co. Ltd.
Email: liuzq@chinatelecom.com.cn

Hailong Bai
China United Network Communications Group Co. Ltd.
Email: baihl@chinaunicom.cn

Juan Zhang
China Mobile Communications Co.,Ltd.
Email: zhangjuan@chinamobile.com

Shuang Li
Alibaba Group Holding Ltd.
Email: shuang.ls@alibaba-inc.com

Xiaohong Shi
Qihoo 360 Technology Co. Ltd.
Email: shixiaohong@360.cn

Yougen Zou
Tencent Holdings Ltd.
Email: living_stone@114dns.com

APPENDIX: Recommendations

As emphasized in the abstract, this survey is motivated from the cooperation of cache service and then the following suggestions areproposed based on the above conclusions, in order to optimize the DNS cache service:

1) Considering the wide deployment of the "Big recursive service" and its impacts mentioned above, a transparent, harmonious and win-win cooperation between authoritative server and recursive server is needed. Typically, authoritative server may provide the recursive server with the latest authoritative data to improve the cache hit-ratio and emergency response ability, and the recursive server may provide the authoritative server the local query statistical data along with a normal NS or zone query as a service optimization factor for the authoritative service operator.

2) Operators individually manage the backup server mainly as an emergency response of the recursive service in the autonomous area. It is suggested that local community should construct and maintain a trusted and shared backup server cooperately, and in this way, the emergency recovery function of the backup servercan cover more recursive services. This trusted and shared backup server is the representative of local community and it is more eligible to build up a more efficient and fluent scheme to manage and collect the backup data.

(We herein only list the main suggestions to coordinate the recursive service. Detailed solution and service architecture will be proposed in the future. Of course, some operators may favor these ideas, but they don’t need to be standardized)

Author’s Address

Wei Wang
NANEL (Naming&Addressing National Engeering Lab)
No.4 South 4th Street, Zhongguancun
Beijing, P. R. China
Email: wangwei@cnnic.cn

Zhiwei Yan
NANEL (Naming&Addressing National Engeering Lab)
No.4 South 4th Street, Zhongguancun
Beijing, P. R. China
Email: yanzhiwei@cnnic.cn

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

dnsop
Internet-Draft
Intended status: Informational
Expires: November 19, 2015

W. Kumari
Google
A. Sullivan
Dyn

May 18, 2015

The ALT Special Use Top Level Domain
draft-wkumari-dnsop-alt-tld-06

Abstract

This document reserves a string (ALT) to be used as a TLD label in non-DNS contexts or for names that have no meaning in a global context. It also provides advice and guidance to developers developing alternate namespaces.

[Ed note: This document lives in GitHub at: https://github.com/wkumari/draft-wkumari-dnsop-alt-tld . Issues and pull requests happily accpeted.]

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Many protocols and systems need to name entities. Names that look like DNS names (a series of labels separated with dots) have become common, even in systems that are not part of the global DNS.

This document provides a solution that may be more appropriate than [RFC6761] in many cases. RFC6761 specifies Special Use TLDs which should only be used in exceptional circumstances.

This document reserves the label "ALT" (short for "Alternate") as a Special Use Domain ([RFC6761]). This label is intended to be used as the final label (apart from the zero-length terminating label) to signify that the name is not rooted in the DNS, and that normal registration and lookup rules do not apply.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

This document assumes familiarity with DNS terms and concepts. Please see [RFC1034] for background and concepts.

- o DNS context: The namespace anchored at the globally-unique DNS root. This is the namespace or context that "normal" DNS uses.

- o non-DNS context: Any other (alternate) namespace.

- o pseudo-TLD: A label that appears in a fully-qualified domain name in the position of a TLD, but which is not registered in the global DNS.

- o TLD: The last visible label in either a fully-qualified domain name or a name that is qualified relative to the root. See the discussion in Section 2.

2. Background

The DNS data model is based on a tree structure, and has a single root. Conventionally, a name immediately beneath the root is called a "Top Level Domain" or "TLD". TLDs usually delegate portions of their namespace to others, who may then delegate further. The hierarchical, distributed and caching nature of the DNS has made it the primary resolution system on the Internet.

Domain names are terminated by a zero-length label, so the root label is normally invisible. Truly fully-qualified names indicate the root label explicitly, thus: "an.example.tld.". Most of the time, names are written implicitly relative to the root, thus: "an.example.tld". In both of these cases, the TLD is the last label that is visible in presentation format -- in this example, the string "tld". (This little bit of pedantry is here because, in different contexts, people can use the term "fully-qualified domain name" to refer to either of these uses.) It is worth noting that the root label is present in the on-wire format of fully-qualified domain names, even if not displayed in the presentation form.

The success of the DNS makes it a natural starting point for systems that need to name entities in a non-DNS context, or that have no unique meaning in a global context. These name resolutions, therefore, occur in a namespace distinct from the DNS.

In many cases, these systems build a DNS-style tree parallel to the global DNS administered by IANA. They often use a pseudo-TLD to cause resolution in the alternate namespace, using browser plugins,

shims in the name resolution process, or simply applications that perform special handling of this alternate namespace.

In many cases, the creators of these alternate namespaces have chosen a convenient or descriptive string and started using it. These new strings are "alternate" strings and are not registered anywhere or part of the DNS. However they appear to be TLDs. Issues may arise if they are looked up in the DNS. These include:

- o User confusion: If someone emails a link of the form foo.bar.pseudo-TLD to someone who does not have the necessary software to resolve names in the pseudo-TLD namespace, the name will not resolve and the user may become confused.

- o Excess traffic hitting the DNS root: Lookups leak out of the pseudo-TLD namespace and end up hitting the DNS root nameservers.

- o Collisions: If the pseudo-TLD is eventually delegated from the root zone the behavior may be non-deterministic.

- o Lack of success for the user’s original goal.

An alternate name resolution system might be specifically designed to provide confidentiality of the looked up name, and to provide a distributed and censorship resistant namespace. This goal would necessarily be defeated if the queries leak into the DNS, because the attempt to look up the name would be visible at least to the operators of root name servers.

3. The ALT namespace

In order to avoid the above issues, we reserve the ALT label. Unless the name desired is globally unique, has meaning on the global context and is delegated in the DNS, it should be considered an alternate namespace, and follow the ALT label scheme outlined below. The ALT label MAY be used in any domain name as a pseudo-TLD to signify that this is an alternate (non-DNS) namespace.

Alternate namespaces should differentiate themselves from other alternate namespaces by choosing a name and using it in the label position just before the pseudo-TLD (ALT). For example, a group wishing to create a namespace for Friends Of Olaf might choose the string "foo" and use any set of labels under foo.alt.

As they are in an alternate namespace, they have no significance in the regular DNS context and so should not be looked up in the DNS context. Unfortunately simply saying that "something should not happen" doesn’t actually stop it from happening, so we need some

rules to guide implementors and operators. The ALT TLD is delegated to "new style" AS112 servers, and so recursive and stub resolvers will get NXDOMAIN for all queries.

1. Iterative resolvers SHOULD follow the advice in [RFC6303], Section 3.

2. The ALT TLD is delegated to "new style" AS112 nameservers ([I-D.ietf-dnsop-as112-dname]), which will return NXDOMAIN for all queries.

These rules are intended to limit how far unintentional queries (i.e. those not intended for the global DNS) flow.

Groups wishing to create new alternate namespaces SHOULD create their alternate namespace under a label that names their namespace, and under the ALT label. They SHOULD choose a label that they expect to be unique and, ideally, descriptive.

Currently deployed projects and protocols that are using pseudo-TLDs may decide to move under the ALT TLD, but this is not a requirement. Rather, the ALT TLD is being reserved so that future projects of a similar nature have a designated place to create alternate resolution namespaces that will not conflict with the regular DNS context.

A number of names other than .ALT were considered and discarded. In order for this technique to be effective the names need to continue to follow both the DNS format and conventions (a prime consideration for alternate name formats is that they can be entered in places that normally take DNS context names); this rules out using suffixes that do not follow the usual letter, digit, and hyphen label convention. Another proposal was that the ALT TLD instead be a reservation under .arpa. This was considered, but rejected for several reasons, including:

1. We wished this to make it clear that this is not in the DNS context, and .arpa clearly is.

2. The use of the string .ALT is intended to evoke the alt.* hierarchy in Usenet.

3. We wanted the string to be short and easily used.

4. A name underneath .arpa would consume at least five additional octets of the total 255 octets available in domain names, which could put pressure on applications that need long machine-generated names.

5. We are suggesting that the string .ALT get special treatment in resolvers, and shim software. We are concerned that using subdomains of an existing TLD (like .arpa) might end up with bad implementations misconfiguring / overriding the TLD itself and breaking .arpa.

There is a concern that if there were placed under .arpa, inexperienced nameserver operators may inadvertently cover .arpa. A more significant concern is that the scope of the issue if the query does leak, and the fact that this would then make the root of the alternate naming namespace a third level domain, and not a second one. A project may be willing to have a name of the form example.alt, but example.alt.arpa may be not look as good.

4. Advice to developers

Often, a subdomain of an existing, owned domain may suffice. When that is so, using a subdomain in the DNS is always preferable, and safest in terms of not risking misuse, duplications, or collisions. In the rare instance in which it is not desirable to have the name in the DNS, the .ALT namespace may be used.

In a number of cases the purpose of the alternate name resolution system is to provide confidentiality. For these systems the above advice is problematic. If the query for one of these names (for example harry.foo.example.com were to leak into the DNS, the query would hit the recursive resolver, and (assuming empty caches) would then hit the root, the .com name servers, the example.com name servers and then the foo.example.com nameservers. This means that the fact that a user is resolving harry.foo.example.com would be visible to a large number of people. Furthermore, the harry.foo.example.com nameservers become a good oracle to determine what names exist, and who is trying to reach them.

For projects that are very latency sensitive, or that desire to provide confidentiality, we recommend anchoring the alternate namespace under the .ALT TLD.

5. IANA Considerations

The IANA is requested to add the ALT string to the "Special-Use Domain Name" registry ([RFC6761], and reference this document. In addition, the "Locally Served DNS Zones" ([RFC6303]) registry should be updated to reference this document.

5.1. Domain Name Reservation Considerations

This section is to satisfy the requirement in Section 5 of RFC6761.

The domain "alt.", and any names falling within ".alt.", are special in the following ways:

1. Human users are expected to know that strings that end in .alt behave differently to normal DNS names. Users are expected to have applications running on their machines that intercept stings of the form <namespace>.alt and perform special handing of them. If the user tries to resolve a name of the form <namespace>.alt without the <namespace> plugin installed, the request will leak into the DNS, and receive a negative response.

2. Writers of application software that implement a non-DNS namespace are expected to intercept names of the form <namespace>.alt and perform application specific handing with them. Other applications are not intended to perform any special handing.

3. In general, writers of name resolution APIs and libraries do not need to perform special handing of these names. If developers of other namespaces implement their namespace through a "shim" or library, they will need to intercept and perform their own handling.

4. Caching DNS servers SHOULD recognize these names as special and SHOULD NOT, by default, attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve these names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.

5. Authoritative DNS servers SHOULD recognize these names as special and SHOULD, by default, generate immediate negative responses for all such queries, unless explicitly configured by the administrator to give positive answers for private-address reverse-mapping names.

6. DNS server operators SHOULD be aware that queries for names ending in .alt are not DNS names, and were leaked into the DNS context (for example, by a missing browser plugin). This information may be useful for support or debuggung purposes.

7. DNS Registries/Registrars MUST NOT grant requests to register "alt" names in the normal way to any person or entity. These "alt" names are defined by protocol specification to be

nonexistent, and they fall outside the set of names available for allocation by registries/registrars.

6. Security Considerations

One of the motivations for the creation of the alt pseudo-TLD is that unmanaged labels in the managed root name space are subject to unexpected takeover if the manager of the root name space decides to delegate the unmanaged label.

The unmanaged and "registration not required" nature of labels beneath .ALT provides the opportunity for an attacker to re-use the chosen label and thereby possibly compromise applications dependent on the special host name.

7. Acknowledgements

The authors understand that there is much politics surrounding the delegation of a new TLD and thank the ICANN liaison in advance.

We would also like to thank Joe Abley, Mark Andrews, Marc Blanchet, John Bond, Stephane Bortzmeyer, David Cake, David Conrad, Patrik Faltstrom, Olafur Gudmundsson, Paul Hoffman, Joel Jaeggli, Ted Lemon, Edward Lewis, George Michaelson, Ed Pascoe, Arturo Servin, and Paul Vixie for feedback.

8. References

8.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, July 2011.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

8.2. Informative References

[I-D.ietf-dnsop-as112-dname] Abley, J., Dickson, B., Kumari, W., and G. Michaelson, "AS112 Redirection using DNAME", draft-ietf-dnsop-as112-dname-06 (work in progress), November 2014.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -05 to -06

- o Incorporated comments from a number of people, including a number of suggestion heard at the IETF meeting in Dallas, and the DNSOP Interim meeting in May, 2015.

- o Removed the "Let’s have an (optional) IANA registry for people to (opportinistically) register their string, if they want that option" stuff. It was, um, optional....

From -04 to -05

- o Went through and made sure that I’d captured the feedback received.

- o Comments from Ed Lewis.

- o Filled in the "Domain Name Reservation Considerations" section of RFC6761.

- o Removed examples from .Onion.

From -03 to -04

- o Incorporated some comments from Paul Hoffman

From -02 to -03

- o After discussions with chairs, made this much more generic (not purely non-DNS), and some cleanup.

From -01 to -02

- o Removed some fluffy wording, tightened up the language some.

From -00 to -01.

- o Fixed the abstract.

- o Recommended that folk root their non-DNS namespace under a DNS namespace that they control (Joe Abley)

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Andrew Sullivan
Dyn
150 Dow Street
Manchester, NH 03101
US

Email: asullivan@dyn.com