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

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 19, 2016.

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

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

12.2. Version 02

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

12.3. Version 03

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

- 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

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 24, 2015

C. Grothoff
INRIA
M. Wachs
Technische Universitaet Muenchen
H. Wolf, Ed.
GNU consensus
J. Appelbaum
L. Ryge
Tor Project Inc.
June 30, 2015

Special-Use Domain Name for Namecoin
draft-grothoff-iesg-special-use-p2p-bit-00

Abstract

This document registers a Special-Use Domain Name for use with the Namecoin system, as per RFC6761.

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 December 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) is primarily used to map human-memorable names to IP addresses, which are used for routing but generally not meaningful for humans.

Namecoin offers a specific timeline-based mechanism to allocate, register, manage, and resolve names, independently from the DNS root and delegation tree.

As compatibility with applications using domain names is desired, Namecoin uses an exclusive alternative Top-Level Domain to avoid conflicts between the Namecoin namespace and the DNS hierarchy.

In order to avoid interoperability issues with DNS as well as to address security and privacy concerns, this document registers the Special-Use Domain Names "BIT" for use with Namecoin, as per [RFC6761].

Namecoin (also known as the Dot-Bit Project) uses this pTLD to realize censorship-resistant naming.

2. Applicability

[RFC6761] Section 3 states:

> "[I]f a domain name has special properties that affect the way hardware and software implementations handle the name, that apply universally regardless of what network the implementation may be connected to, then that domain name may be a candidate for having

> the IETF declare it to be a Special-Use Domain Name and specify what special treatment implementations should give to that name. On the other hand, if declaring a given name to be special would result in no change to any implementations, then that suggests that the name may not be special in any material way, and it may be more appropriate to use the existing DNS mechanisms [RFC1034] to provide the desired delegation, data, or lack-of-data, for the name in question. Where the desired behaviour can be achieved via the existing domain name registration processes, that process should be used. Reservation of a Special-Use Domain Name is not a mechanism for circumventing normal domain name registration processes."

The Special-Use Domain Name for Namecoin reserved by this document meets this requirement, as it has the following specificities:

- The "BIT" pTLD is not manageable by some designated administration. Instead, it is managed by a P2P protocol using a global public ledger.
- Namecoin does not depend on the DNS context for their resolution: Namecoin domains MAY use the DNS servers infrastructure, as they return DNS-compatible results; but it uses specific P2P protocols for regular name resolution, covered by the respective protocol specifications.
- When Namecoin is properly implemented, the implementation MUST intercept queries for the pTLD to ensure Namecoin names cannot leak into the DNS.
- The appropriate pTLD protocols can be implemented in existing software libraries and APIs to extend regular DNS operation and enable Namecoin name resolution. However, the default hierarchical DNS response to any request to any pTLD MUST be NXDOMAIN.
- Finally, in order for Namecoin to realize a censorship-resistant name system, this document specifies changes required in existing DNS software and DNS operations.

3. Terminology and Conventions Used in This Document

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].

The word "peer" is used in the meaning of a individual system on the network.

The abbreviation "pTLD" is used in this document to mean a pseudo Top-Level Domain, i.e., a Special-Use Domain Name per [RFC6761] reserved to P2P Systems in this document. A pTLD is mentioned in capitals, and within double quotes to mark the difference with a regular DNS gTLD.

In this document, ".tld" (lowercase, with quotes) means: any domain or hostname within the scope of a given pTLD, while .tld (lowercase, without quotes) refers to an adjective form. For example, a collection of ".bit" peers in "BIT", but an .bit URL. [TO REMOVE: in the IANA Considerations section, we use the simple .tld format to request TLD reservation for consistency with previous RFCs].

The word "NXDOMAIN" refers to an alternate expression for the "Name Error" RCODE as described in section 4.1.1 of [RFC1035]. When referring to "NXDOMAIN" and negative caching [RFC2308] response, this document means an authoritative (AA=1) name error (RCODE=3) response exclusively.

4. The "BIT" Timeline System pTLD

Namecoin is a timeline-based system in the style of Bitcoin to create a global, secure, and memorable name system. It creates a single, globally accessible, append-only timeline of name registrations. Timeline-based systems rely on a peer-to-peer network to manage updates and store the timeline. In the Namecoin system, modifications to key-value mapping are attached to transactions which are committed to the timeline by "mining". Mining is a proof-of-work calculation that uses brute-force methods to find (partial) hash collisions with a state summary (fingerprint) representing the complete global state -- including the full history -- of the timeline .

"BIT" provides a name space where names are registered via transactions in the Namecoin currency [Namecoin]. Like Bitcoins, Namecoins are used to establish a decentralized, multi-party consensus on the valid transaction history, and thus the set of registered names and their values [SquareZooko].

The Namecoin used in a transaction to register a name in "BIT" is lost. This is not a fundamental problem as more coins can be generated via mining (proof-of-work calculations). The registration cost is set to decrease over time, to prevent early adopters from registering too many names.

The owner of a name can update the associated value by issuing an update, which is a transaction that uses a special coin. This coin

is generated as change during the registration operation. If a name is not updated for a long time, the registration expires.

Performing a lookup for a name with Namecoin consists in checking the timeline for correctness to ensure the validity of the blockchain, and traversing it to see if it contains an entry for the desired name. Namecoin supports resolution for other peer-to-peer systems such as ".onion" and ".i2p" via specific resource records.

Like DNS registry, the Dot-Bit registry is public. But unlike DNS, the public registry is maintained by network consensus on the blockchain. It departs from DNS in three ways:

first, domain names are not delegated to an authority that can assign them, but acquired by the operating party (the "domain owner"), in the form of a historical claim made directly by appending to the Namecoin blockchain. The domain is thus bound not to a legal contract with an administrative authority, but to a cryptographic coin, and the network consensus on the timeline.

second, the timeline contains the entire registry for all .bit domains: the Namecoin blockchain itself is the complete domain database. As participant peers maintain the consensus on the timeline, they store a local copy of the Namecoin blockchain. Therefore, to those peers, name resolution and registry traversal are both local and private. Each participant theoretically has the whole domain’s database. In practice, some users can trust a name server to access the Namecoin blockchain on their behalf.

third, the Namecoin system is not limited to domain names and can store arbitrary data types. Each record must follow the same rules (expiry time, data size limits, etc.). The Namecoin’s Domain Name Specification [Namecoin-DNS] defines the "d namespace" for use with "BIT" and other unrelated namespaces co-exist on the Namecoin blockchain.

The "BIT" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

   From the user’s perspective, the resolution of .bit names is similar to the normal DNS resolution, and thus should not affect normal usage of most Internet applications.

2. Application software SHOULD NOT recognize .bit domains as special and SHOULD treat them as they would other domains.

   Applications MAY pass requests to the "BIT" pTLD to DNS resolvers and libraries if A/AAAA records are desired. If available, the local resolver can intercept such requests within the respective operating system hooks and return DNS-compatible results.

   Namecoin-aware applications MAY choose to talk directly to the respective P2P resolver, and use this to access additional record types that are not defined in DNS.

3. Name resolution APIs and libraries SHOULD either respond to requests for .bit names by resolving them via the Namecoin protocol, or respond with NXDOMAIN.

4. Caching DNS servers SHOULD recognize .bit names as special and SHOULD NOT attempt to resolve them. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.

   Given that .bit users typically have no special privacy expectations, and those names are globally unique, local caching DNS servers MAY choose to treat them as regular domain names, and cache the responses obtained from the Namecoin blockchain. In that case however, NXDOMAIN results SHOULD NOT be cached, as new .bit domains may become active at any time.

5. Authoritative DNS servers are not expected to treat .bit domain requests specially. In practice, they MUST answer with NXDOMAIN, as "BIT" is not available via global DNS resolution.

6. DNS server operators SHOULD be aware that .bit names are reserved for use with Namecoin, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .bit names. This helps avoid conflicts [SAC45]. These names are defined by the Namecoin protocol specification, and they fall outside the set of names available for allocation by registries/ registrars.

5. Security Considerations

Specific software performs the resolution of Namecoin Special-Use Domain Names presented in this document; this resolution process happens outside of the scope of DNS. Leakage of requests to such domains to the global operational DNS can cause interception of traffic that might be misused to monitor, censor, or abuse the user's trust, and lead to privacy issues with potentially tragic consequences for the user.

This document reserves these Top-Level Domain names to minimize the possibility of confusion, conflict, and especially privacy risks for users.

In the introduction of this document, there's a requirement that DNS operators do not override resolution of the Namecoin names. This is a regulatory measure and cannot prevent such malicious abuse in practice. Its purpose is to limit any information leak that would result from incorrectly configured systems, and to avoid that resolvers make unnecessary contact to the DNS Root Zone for such domains. Verisign, Inc., as well as several Internet service providers (ISPs) have notoriously abused their position to override NXDOMAIN responses to their customers in the past [SSAC-NXDOMAIN-Abuse]. For example, if a DNS operator would decide to override NXDOMAIN and send advertising to leaked .onion sites, the information leak to the DNS would extend to the advertising server, with unpredictable consequences. Thus, implementors should be aware that any positive response coming from DNS must be considered with extra care, as it suggests a leak to DNS has been made, contrary to user's privacy expectations.

The reality of X.509 Certificate Authorities (CAs) creating misleading certificates for these pTLDs due to ignorance stresses the need to document their special use. X.509 Certificate Authorities MAY create certificates for "BIT", given CSRs signed with the respective private keys corresponding to the respective names. For "BIT", the Certificate Authority SHOULD limit the expiration time of the certificate to match the registration.

Because the Namecoin system uses a timeline-based blockchain for name assignment and resolution, it grants query privacy to the users who maintain their own copy of the blockchain (Section 4.4), but the entire zone of a .bit domains is publicly available in the Namecoin blockchain, making enumeration of names within a .bit zone ("zone walking") a trivial attack to conduct. This might be a concern to some domain operators as it exposes their infrastructure to potential adversaries. That concern may be addressed in future versions of Namecoin, but the records already in the blockchain will remain there unprotected.

Finally, legacy applications that do not explicitly support the Namecoin pTLD significantly increase the risk of ".bit" queries escaping to DNS, as they are entirely dependent on the correct configuration on the operating system.

6. IANA Considerations

The Internet Assigned Numbers Authority (IANA) reserved the following entries in the Special-Use Domain Names registry [RFC6761]:

.bit

[TO REMOVE: the assignement URL is https://www.iana.org/assignments/special-use-domain-names/]

7. Acknowledgements

The authors thank the I2P and Namecoin developers for their constructive feedback, as well as Mark Nottingham for his proof-reading and valuable feedback. The authors also thank the members of DNSOP WG for their critiques and suggestions.

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.

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

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

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

8.2. Informative References

[Namecoin]
The .bit Project, "Namecoin", 2013, <https://namecoin.org/>.

[Namecoin-DNS]
The .bit Project, "Namecoin Domain Name Specification", 2015, <https://bit.namecoin.org/spec>.

[SAC45] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System", November 2010, <http://www.icann.org/en/groups/ssac/documents/sac-045-en.pdf>.

[SquareZooko]
Swartz, A., "Squaring the Triangle: Secure, Decentralized, Human-Readable Names", 2011, <http://www.aaronsw.com/weblog/squarezooko>.

[SSAC-NXDOMAIN-Abuse]
ICANN Security and Stability Advisory Committee, "Redirection in the COM and NET Domains", July 2004, <http://www.icann.org/committees/security/ssac-report-09jul04.pdf>.

Authors' Addresses

Christian Grothoff
INRIA
Equipe Decentralisee
INRIA Rennes Bretagne Atlantique
263 avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes, Bretagne F-35042
FR

Email: christian@grothoff.org

Matthias Wachs
Technische Universitaet Muenchen
Free Secure Network Systems Group
Lehrstuhl fuer Netzarchitekturen und Netzdienste
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: wachs@net.in.tum.de

Hellekin O. Wolf (editor)
GNU consensus

Email: hellekin@gnu.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Leif Ryge
Tor Project Inc.

Email: leif@synthesize.us

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 24, 2015

C. Grothoff
INRIA
M. Wachs
Technische Universitaet Muenchen
H. Wolf, Ed.
GNU consensus
J. Appelbaum
L. Ryge
Tor Project Inc.
June 30, 2015

The .exit Special-Use Domain Name of Tor
draft-grothoff-iesg-special-use-p2p-exit-00

Abstract

This document registers a Special-Use Domain Name for use with the Tor Project, as per RFC6761.

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 December 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) is primarily used to map human-memorable names to IP addresses, which are used for routing but generally not meaningful for humans.

The Tor project supports the use of names to specify where the user wishes to exit the P2P overlay.

As compatibility with applications using domain names is desired, this mechanism requires an exclusive alternative Top-Level Domains to avoid conflict between the Tor namespace and the DNS hierarchy.

In order to avoid interoperability issues with DNS as well as to address security and privacy concerns, this document registers the "EXIT" Special-Use Domain Names for use within the Tor network, as per [RFC6761].

The Tor network uses this pTLD to control overlay routing and to securely specify path selection choices [TOR-PATH].

2. Applicability

[RFC6761] Section 3 states:

> "[I]f a domain name has special properties that affect the way hardware and software implementations handle the name, that apply universally regardless of what network the implementation may be connected to, then that domain name may be a candidate for having the IETF declare it to be a Special-Use Domain Name and specify

what special treatment implementations should give to that name. On the other hand, if declaring a given name to be special would result in no change to any implementations, then that suggests that the name may not be special in any material way, and it may be more appropriate to use the existing DNS mechanisms [RFC1034] to provide the desired delegation, data, or lack-of-data, for the name in question. Where the desired behaviour can be achieved via the existing domain name registration processes, that process should be used. Reservation of a Special-Use Domain Name is not a mechanism for circumventing normal domain name registration processes."

The set "EXIT" pTLD reserved by this document meets this requirement, as it has the following specificities:

- o "EXIT" resolution does not depend on the DNS context: The name specifies a Tor exit node, and thus the response is not even really DNS-compatible; Tor uses its own P2P protocols for resolving the destination specified in an .exit name.

- o When Tor is properly implemented, the implementation MUST intercept queries for the "EXIT" to ensure that these Tor-specific names cannot leak into the DNS.

- o Finally, in order for Tor to properly interoperate with DNS and to provide security and privacy features matching user expectations, this document specifies desirable changes in existing DNS software and DNS operations.

3. Terminology and Conventions Used in This Document

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].

The word "peer" is used in the meaning of a individual system on the network.

The abbreviation "pTLD" is used in this document to mean a pseudo Top-Level Domain, i.e., a Special-Use Domain Name per [RFC6761] reserved to P2P Systems in this document. A pTLD is mentioned in capitals, and within double quotes to mark the difference with a regular DNS gTLD.

In this document, ".tld" (lowercase, with quotes) means: any domain or hostname within the scope of a given pTLD, while .tld (lowercase, without quotes) refers to an adjective form.

The word "NXDOMAIN" refers to an alternate expression for the "Name Error" RCODE as described in section 4.1.1 of [RFC1035]. When referring to "NXDOMAIN" and negative caching [RFC2308] response, this document means an authoritative (AA=1) name error (RCODE=3) response exclusively.

The Tor-related names such as 'circuit', 'exit', 'node', 'relay', 'stream', and related Tor terms are described in [Dingledine2004] and the Tor protocol specification [TOR-PROTOCOL].

4. The "EXIT" Client Source Routing pTLD

The .exit suffix is used as an in-band source routing control channel, usually for selection of a specific Tor relay during path creation as the last node in the Tor circuit.

It may be used to access a DNS host via specific Torservers, in the form "hostname.nickname-or-fingerprint.exit", where the "hostname" is a valid hostname, and the "nickname-or-fingerprint" is either the nickname of a Tor relay in the Tor network consensus, or the hex-encoded SHA1 digest of the given node's public key (fingerprint).

For example, "gnu.org.noisetor.exit" will route the client to "gnu.org" via the Tor node nicknamed "noisetor". Using the fingerprint instead of the nickname ensures that the path selection uses a specific Tor exit node, and is harder to remember: e.g., "gnu.org.f97f3b153fed6604230cd497a3d1e9815b007637.exit".

When Tor sees an address in this format, it uses the specified "nickname-or-fingerprint" as the exit node. If no "hostname" component is given, Tor defaults to the published IPv4 address of the Tor exit node [TOR-EXTSOCKS].

Because "hostname" is allegedly valid, the total length of a .exit construct may exceed the maximum length allowed for domain names. Moreover, the resolution of "hostname" happens at the exit node. Trying to resolve such invalid domain names, including chaining .exit names will likely return a DNS lookup failure at the first exit node.

The "EXIT" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

   Since .exit names correspond to a Tor-specific routing construct to reach target hosts via chosen Tor exit nodes, users need to be

aware that they do not belong to regular DNS and that the actual target precedes the second-level domain name.

2. Application software MAY recognize that .exit domains are special and when they do SHOULD NOT pass requests for these domains to DNS resolvers and libraries.

   As mentioned in items 4 and 5 below, regular DNS resolution is expected to respond with NXDOMAIN. Therefore, if it can differentiate between DNS and P2P name resolution, application software:

   * MUST expect NXDOMAIN as the only valid DNS response, and

   * SHOULD treat other answers from DNS as errors.

   Tor-aware applications MAY also use Tor resolvers directly.

3. Name resolution APIs and libraries SHOULD either respond to requests for .exit names by resolving them via the Tor protocol, or respond with NXDOMAIN.

4. Caching DNS servers SHOULD recognize .exit 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 .exit names. Instead, caching DNS servers SHOULD, by default, generate immediate negative responses for all such queries.

5. Authoritative DNS servers are not expected to treat .exit domain requests specially. In practice, they MUST answer with NXDOMAIN, as "EXIT" is not available via global DNS resolution, and not doing so MAY put users’ privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .exit names are reserved for use with Tor, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .exit names. This helps avoid conflicts [SAC45]. These names are defined by the Tor address specification, and they fall outside the set of names available for allocation by registries/ registrars.

5. Security Considerations

Specific software performs the resolution of the six Special-Use Domain Names presented in this document; this resolution process happens outside of the scope of DNS. Leakage of requests to such domains to the global operational DNS can cause interception of traffic that might be misused to monitor, censor, or abuse the user's trust, and lead to privacy issues with potentially tragic consequences for the user.

This document reserves these Top-Level Domain names to minimize the possibility of confusion, conflict, and especially privacy risks for users.

In the introduction of this document, there's a requirement that DNS operators do not override resolution of the "EXIT" Names. This is a regulatory measure and cannot prevent such malicious abuse in practice. Its purpose is to limit any information leak that would result from incorrectly configured systems, and to avoid that resolvers make unnecessary contact to the DNS Root Zone for such domains. Verisign, Inc., as well as several Internet service providers (ISPs) have notoriously abused their position to override NXDOMAIN responses to their customers in the past [SSAC-NXDOMAIN-Abuse]. For example, if a DNS operator would decide to override NXDOMAIN and send advertising to leaked .onion sites, the information leak to the DNS would extend to the advertising server, with unpredictable consequences. Thus, implementors should be aware that any positive response coming from DNS must be considered with extra care, as it suggests a leak to DNS has been made, contrary to user's privacy expectations.

The reality of X.509 Certificate Authorities (CAs) creating misleading certificates for these pTLDs due to ignorance stresses the need to document their special use. Certificate Authorities MUST NOT create certificates for "EXIT" Top-level domains. Nevertheless, clients SHOULD accept certificates for these Top-Level domains as they may be created legitimately by local proxies on the fly.

Finally, legacy applications that do not explicitly support the pTLD significantly increase the risk of pTLD queries escaping to DNS, as

they are entirely dependent on the correct configuration on the operating system.

6. IANA Considerations

The Internet Assigned Numbers Authority (IANA) reserved the following entries in the Special-Use Domain Names registry [RFC6761]:

.exit

[TO REMOVE: the assignement URL is https://www.iana.org/assignments/special-use-domain-names/]

7. Acknowledgements

The authors thank the I2P and Namecoin developers for their constructive feedback, as well as Mark Nottingham for his proof-reading and valuable feedback. The authors also thank the members of DNSOP WG for their critiques and suggestions.

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.

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

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

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

8.2. 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>.

[SAC45] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System", November 2010, <http://www.icann.org/en/groups/ssac/documents/sac-045-en.pdf>.

[SSAC-NXDOMAIN-Abuse] ICANN Security and Stability Advisory Committee, "Redirection in the COM and NET Domains", July 2004, <http://www.icann.org/committees/security/ssac-report-09jul04.pdf>.

[TOR-EXTSOCKS] Mathewson, N. and R. Dingledine, "Tor's extensions to the SOCKS protocol", February 2014, <https://gitweb.torproject.org/torspec.git/plain/socks-extensions.txt>.

[TOR-PATH] Mathewson, N. and R. Dingledine, "Tor Path Specification", November 2014, <https://gitweb.torproject.org/torspec.git/plain/path-spec.txt>.

[TOR-PROTOCOL] Dingledine, R. and N. Mathewson, "Tor Protocol Specification", August 2014, <https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt>.

Authors' Addresses

Christian Grothoff
INRIA
Equipe Decentralisee
INRIA Rennes Bretagne Atlantique
263 avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes, Bretagne F-35042
FR

Email: christian@grothoff.org

Matthias Wachs
Technische Universitaet Muenchen
Free Secure Network Systems Group
Lehrstuhl fuer Netzarchitekturen und Netzdienste
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: wachs@net.in.tum.de

Hellekin O. Wolf (editor)
GNU consensus

Email: hellekin@gnu.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Leif Ryge
Tor Project Inc.

Email: leif@synthesize.us

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 24, 2015

C. Grothoff
INRIA
M. Wachs
Technische Universitaet Muenchen
H. Wolf, Ed.
GNU consensus
J. Appelbaum
L. Ryge
Tor Project Inc.
June 30, 2015

Special-Use Domain Names of the GNU Name System
draft-grothoff-iesg-special-use-p2p-gns-00

Abstract

This document registers a set of Special-Use Domain Names for use with Peer-to-Peer (P2P) systems, as per RFC6761.

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 December 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

The GNU Name System (GNS) uses "GNU" and "ZKEY" to realize privacy-enhanced, fully-decentralized and censorship-resistant naming.

In order to avoid interoperability issues with DNS as well as to address security and privacy concerns, this document registers a set of Special-Use Domain Names for use with P2P systems (pTLDs), as per [RFC6761],: "GNU" and "ZKEY".

2. Applicability

[RFC6761] Section 3 states:

> "[I]f a domain name has special properties that affect the way hardware and software implementations handle the name, that apply universally regardless of what network the implementation may be connected to, then that domain name may be a candidate for having the IETF declare it to be a Special-Use Domain Name and specify what special treatment implementations should give to that name. On the other hand, if declaring a given name to be special would result in no change to any implementations, then that suggests that the name may not be special in any material way, and it may be more appropriate to use the existing DNS mechanisms [RFC1034] to provide the desired delegation, data, or lack-of-data, for the name in question. Where the desired behaviour can be achieved via the existing domain name registration processes, that process should be used. Reservation of a Special-Use Domain Name is not a

mechanism for circumventing normal domain name registration processes."

The set of Special-Use Domain Names for the GNU Name System (pTLDs) reserved by this document meet this requirement, as they share the following specificities:

- pTLDs are not manageable by some designated administration. Instead, they are managed according to various alternate strategies or combinations thereof, introduced in this document, and their respective protocol specifications: automated cryptographic assignment (".zkey"), or user-controled assignment in a private scope (".gnu").

- The pTLDs do not depend on the DNS context for their resolution: GNS resolution MAY involve the DNS server infrastructure, as it returns DNS-compatible results; however, a specific P2P protocol is used for regular name resolution, covered by its respective protocol specification.

- GNS name resolution is typically integrated with existing software libraries and APIs to extend regular DNS operation and enable more secure name resolution. GNS implementations MUST intercept queries for the respective pTLDs to ensure GNS names cannot leak into the DNS from properly configured systems. Nevertheless, in case GNS names do leak into the DNS, the default hierarchical DNS response to any request to any pTLD MUST be NXDOMAIN.

- Finally, in order to facilitate the GNU Name System’s vision of a censorship-resistant, fully-decentralized name system, and provide security and privacy features matching user expectations, this document specifies desirable changes in existing DNS software and DNS operations.

3. Terminology and Conventions Used in This Document

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].

The word "peer" is used in the meaning of a individual system on the network.

The abbreviation "pTLD" is used in this document to mean a pseudo Top-Level Domain, i.e., a Special-Use Domain Name per [RFC6761] reserved to the GNU Name System in this document. A pTLD is mentioned in capitals, and within double quotes to mark the difference with a regular DNS gTLD.

In this document, ".tld" (lowercase, with quotes) means: any domain or hostname within the scope of a given pTLD, while .tld (lowercase, without quotes) refers to an adjective form. For example, a collection of ".gnu" peers in "GNU", but an .gnu URL. [TO REMOVE: in the IANA Considerations section, we use the simple .tld format to request TLD reservation for consistency with previous RFCs].

The word "NXDOMAIN" refers to an alternate expression for the "Name Error" RCODE as described in section 4.1.1 of [RFC1035]. When referring to "NXDOMAIN" and negative caching [RFC2308] response, this document means an authoritative (AA=1) name error (RCODE=3) response exclusively.

4. Description of Special-Use Domains in P2P Networks

4.1. The "GNU" Relative pTLD

"GNU" is used to specify that a domain name should be resolved using GNS. The GNS resolution process is documented in [Wachs2014].

The "GNU" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

   Since there is no central authority responsible for assigning .gnu names, and that specific domain is local to the local peer, users need to be aware of that specificity.

   Legacy applications MAY expect the DNS-to-GNS proxy to return DNS compatible results for the resolution of .gnu domains.

2. Legacy application software does not need to recognize .gnu domains as special, and may continue to use these names as they would other domain names.

   GNS-aware applications MAY also use GNS resolvers directly to resolve .gnu domains (in particular, if they want access to GNS-specific record types).

3. Name resolution APIs and libraries SHOULD either respond to requests for .gnu names by resolving them via the GNS protocol, or respond with NXDOMAIN.

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

5. Authoritative DNS servers are not expected to treat .gnu domain requests specially. In practice, they MUST answer with NXDOMAIN, as "GNU" is not available via global DNS resolution, and not doing so can put users' privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .gnu names are reserved for use with GNS, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .gnu names. This helps avoid conflicts [SAC45]. These names are defined by the GNS protocol specification, and they fall outside the set of names available for allocation by registries/ registrars.

4.2. The "ZKEY" Compressed Public Key pTLD

The "ZKEY" pTLD is used to signify that resolution of the given name MUST be performed using a record signed by an authority that is in possession of a particular public key. Names in "ZKEY" MUST end with a domain which is the compressed point representation from [EdDSA] on [Curve25519] of the public key of the authority, encoded using Crockford's variant of base32hex [RFC4648] (with additionally 'U' being considered equal to 'V') for easier optical character recognition. A GNS resolver uses the key to locate a record signed by the respective authority.

"ZKEY" provides a (reverse) mapping from globally unique hashes to public key, therefore .zkey names are non-memorable, and are expected to be hidden from the user [Wachs2014].

The "ZKEY" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

   Since there is no central authority necessary or possible for assigning .zkey names, and those names match cryptographic keys, users need to be aware that they do not belong to regular DNS, but are still global in their scope.

   Legacy applications MAY expect the DNS-to-GNS proxy to return DNS-compatible results for the resolution of .zkey domains.

2. Application software does not need to recognize .zkey domains as special, and may continue to use these names as they would other domain names.

   GNS-aware applications MAY also use GNS resolvers directly to resolve .zkey domains

3. Name resolution APIs and libraries SHOULD either respond to requests for .zkey names by resolving them via the GNS protocol, or respond with NXDOMAIN.

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

5. Authoritative DNS Servers are not expected to treat .zkey domain requests specially. In practice, they MUST answer with NXDOMAIN, as "ZKEY" is not available via global DNS resolution, and not doing so MAY put users' privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .zkey names are reserved for use with GNS, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .zkey names. This helps avoid conflicts [SAC45]. These names are defined as described above, and they fall outside the set of names available for allocation by registries/registrars.

5. Security Considerations

Specific software performs the resolution of names in the GNU Name System; this resolution process happens outside of the scope of DNS. Leakage of requests to such domains to the global operational DNS can cause interception of traffic that might be misused to monitor, censor, or abuse the user's trust, and lead to privacy issues with potentially tragic consequences for the user.

This document reserves these Top-Level Domain names to minimize the possibility of confusion, conflict, and especially privacy risks for users.

In the introduction of this document, there's a requirement that DNS operators do not override resolution of the GNS names. This is a regulatory measure and cannot prevent such malicious abuse in practice. Its purpose is to limit any information leak that would result from incorrectly configured systems, and to avoid that resolvers make unnecessary contact to the DNS Root Zone for such domains. Verisign, Inc., as well as several Internet service providers (ISPs) have notoriously abused their position to override NXDOMAIN responses to their customers in the past [SSAC-NXDOMAIN-Abuse]. For example, if a DNS operator would decide to override NXDOMAIN and send advertising to leaked .zkey sites, the information leak to the DNS would extend to the advertising server, with unpredictable consequences. Thus, implementors should be aware that any positive response coming from DNS must be considered with extra care, as it suggests a leak to DNS has been made, contrary to user's privacy expectations.

The reality of X.509 Certificate Authorities (CAs) creating misleading certificates for these pTLDs due to ignorance stresses the need to document their special use. X.509 Certificate Authorities MAY create certificates for "ZKEY" given CSRs signed with the respective private keys corresponding to the respective names. Certificate Authorities MUST NOT create certificates for "GNU" Top-Level domains. Nevertheless, clients SHOULD accept certificates for "GNU" Top-Level domains as they may be created legitimately by local proxies on the fly.

Finally, legacy applications that do not explicitly support the pTLDs significantly increase the risk of pTLD queries escaping to DNS, as they are entirely dependent on the correct configuration on the operating system.

6. IANA Considerations

The Internet Assigned Numbers Authority (IANA) reserved the following entries in the Special-Use Domain Names registry [RFC6761]:

.gnu

.zkey

[TO REMOVE: the assignement URL is https://www.iana.org/assignments/special-use-domain-names/]

7. Acknowledgements

The authors thank the I2P and Namecoin developers for their constructive feedback, as well as Mark Nottingham for his proof-reading and valuable feedback. The authors also thank the members of DNSOP WG for their critiques and suggestions.

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.

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

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

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

8.2. Informative References

[Curve25519]
Bernstein, D., "Curve25519: new Diffie-Hellman speed record", February 2006, <http://cr.yp.to/ecdh/curve25519-20060209.pdf>.

[EdDSA] Bernstein, D., Duif, N., Lange, T., Schwabe, P., and Y. Yang, "High-speed, high-security signatures", September 2011, <http://ed25519.cr.yp.to/ed25519-20110926.pdf>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[SAC45] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System", November 2010, <http://www.icann.org/en/groups/ssac/documents/sac-045-en.pdf>.

[SSAC-NXDOMAIN-Abuse]
ICANN Security and Stability Advisory Committee, "Redirection in the COM and NET Domains", July 2004, <http://www.icann.org/committees/security/ssac-report-09jul04.pdf>.

[Wachs2014]
Wachs, M., Schanzenbach, M., and C. Grothoff, "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System", October 2014, <https://gnunet.org/gns-paper>.

Authors' Addresses

Christian Grothoff
INRIA
Equipe Decentralisee
INRIA Rennes Bretagne Atlantique
263 avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes, Bretagne F-35042
FR

Email: christian@grothoff.org

Matthias Wachs
Technische Universitaet Muenchen
Free Secure Network Systems Group
Lehrstuhl fuer Netzarchitekturen und Netzdienste
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: wachs@net.in.tum.de

Hellekin O. Wolf (editor)
GNU consensus

Email: hellekin@gnu.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Leif Ryge
Tor Project Inc.

Email: leif@synthesize.us

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 26, 2015

C. Grothoff
INRIA
M. Wachs
Technische Universitaet Muenchen
H. Wolf, Ed.
GNU consensus
J. Appelbaum
L. Ryge
Tor Project Inc.

June 30, 2015

Special-Use Domain Names for I2P
draft-grothoff-iesg-special-use-p2p-i2p-00

Abstract

This document registers a Special-Use Domain Name for use with the I2P Peer-to-Peer system, as per RFC6761.

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 December 26, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) is primarily used to map human-memorable names to IP addresses, which are used for routing but generally not meaningful for humans.

The Invisible Internet Project (I2P) Peer-to-Peer (P2P) system uses a specific decentralized mechanism to allocate, register, manage, and resolve names. The I2P Name System operates entirely outside of DNS, independently from the DNS root and delegation tree.

As compatibility with applications using domain names is desired, the I2P overlay network defines an exclusive alternative Top-Level Domain to avoid conflict between the I2P namespace and the DNS hierarchy.

In order to avoid interoperability issues with DNS as well as to address security and privacy concerns, this document registers the "I2P" Special-Use Domain Names for use with the I2P systems.

I2P uses this pTLD to realize fully-decentralized and censorship-resistant naming.

2. Applicability

[RFC6761] Section 3 states:

> "[I]f a domain name has special properties that affect the way hardware and software implementations handle the name, that apply universally regardless of what network the implementation may be connected to, then that domain name may be a candidate for having

> the IETF declare it to be a Special-Use Domain Name and specify what special treatment implementations should give to that name. On the other hand, if declaring a given name to be special would result in no change to any implementations, then that suggests that the name may not be special in any material way, and it may be more appropriate to use the existing DNS mechanisms [RFC1034] to provide the desired delegation, data, or lack-of-data, for the name in question. Where the desired behaviour can be achieved via the existing domain name registration processes, that process should be used. Reservation of a Special-Use Domain Name is not a mechanism for circumventing normal domain name registration processes."

The Special-Use Domain Name for the I2P System (pTLDs) reserved by this document meets this requirement, as it has the following specificities:

- The "I2P" pTLD is not manageable by some designated administration. Instead, it is managed according to various alternate strategies as described in the I2P documentation.
- The "I2P" pTLD does not depend on the DNS context for its resolution. It uses I2P-specific logic for name resolution, covered by the respective system documentation.
- To resolve "I2P" names, the implementation MUST intercept queries for the pTLD to ensure I2P names cannot leak into the DNS.
- The appropriate resolution procedure can be implemented in existing software libraries and APIs to extend regular DNS operation and enable I2P name resolution. However, the default hierarchical DNS response to any request to any pTLD MUST be NXDOMAIN.
- Finally, in order to maximally protect the security and privacy expectation of I2P users, this document specifies desirable changes in existing DNS software and DNS operations.

3. Terminology and Conventions Used in This Document

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].

The word "peer" is used in the meaning of a individual system on the network.

The abbreviation "pTLD" is used in this document to mean a pseudo Top-Level Domain, i.e., a Special-Use Domain Name per [RFC6761] reserved to P2P Systems in this document. A pTLD is mentioned in capitals, and within double quotes to mark the difference with a regular DNS gTLD.

In this document, ".tld" (lowercase, with quotes) means: any domain or hostname within the scope of a given pTLD, while .tld (lowercase, without quotes) refers to an adjective form. For example, a collection of ".i2p" peers in "I2P", but an .i2p URL. [TO REMOVE: in the IANA Considerations section, we use the simple .tld format to request TLD reservation for consistency with previous RFCs].

The word "NXDOMAIN" refers to an alternate expression for the "Name Error" RCODE as described in section 4.1.1 of [RFC1035]. When referring to "NXDOMAIN" and negative caching [RFC2308] response, this document means an authoritative (AA=1) name error (RCODE=3) response exclusively.

4. The "I2P" Addressbook pTLD

"I2P" provides accessibility to hidden services within the I2P network [zzz2009]. I2P is a scalable, self-organizing, resilient packet switched anonymous network layer, upon which any number of different anonymity or security-conscious applications can operate, using any protocol.

I2P hidden services and clients are identified by Destinations, anonymous analogues of IP addresses. The "I2P" pTLD, chosen in 2003 [I2P-CHOICE], houses two methods for looking up Destinations:

A local table called the addressbook stores a map of .i2p addresses to Destinations. Each user maintains their own mappings that can be shared with others, allowing them to "discover" new names by importing published addressbooks of peers, and they can emulate traditional DNS by choosing to treat these peers as name servers. The comparison however stops here, as only local uniqueness is mandated. As the system is decentralized, "example.i2p" may resolve differently for different peers depending on the state of their respective addressbooks.

To address globally unique names, the I2P developers dedicated the "B32.I2P" subdomain to hold Base32-encoded [RFC4648] references to Destinations. Like .onion addresses, .b32.i2p addresses are self-authenticating. The details of the encoding are out of scope for this document, and documented in [I2P-NAMING]. The purpose of .b32.i2p addresses is similar to ".zkey", that is to enable

(reverse) mapping for a globally unique hidden service that may not have a defined entry in the local addressbook.

The "I2P" domain is special in the following ways:

1. Users can use these names as they would other domain names, entering them anywhere that they would otherwise enter a conventional DNS domain name.

   Since there is no central authority responsible for assigning .i2p names, and that the ultimate mapping is decided by the local peer, users need to be aware of that specificity.

2. Application software SHOULD recognize .i2p domains as special and SHOULD NOT use them as they would other domains.

   Applications SHOULD NOT pass requests for .i2p domains to DNS resolvers and libraries.

   As mentioned in points 4 and 5 below, regular DNS resolution is expected to respond with NXDOMAIN. Therefore, if it can differentiate between DNS and P2P name resolution, application software can expect such a response, and can choose to treat other responses from resolvers and libraries as errors.

3. Name resolution APIs and libraries SHOULD either respond to requests for .i2p names by resolving them via the I2P protocol, or respond with NXDOMAIN.

4. Caching DNS servers SHOULD recognize .i2p names as special and SHOULD NOT attempt to look up NS records for them, or otherwise query authoritative DNS servers in an attempt to resolve .i2p names. Instead, caching DNS servers SHOULD generate immediate negative responses for all such queries.

5. Authoritative DNS servers are not expected to treat .i2p domain requests specially. In practice, they MUST answer with NXDOMAIN, as "I2P" is not available via global DNS resolution, and not doing so MAY put users’ privacy at risk (see item 6).

6. DNS server operators SHOULD be aware that .i2p names are reserved for use with I2P, and MUST NOT override their resolution (e.g., to redirect users to another service or error information).

7. DNS registries/registrars MUST NOT grant any request to register .i2p names. This helps avoid conflicts [SAC45]. These names are defined by the I2P protocol specification, and they fall outside the set of names available for allocation by registries/ registrars.

5. Security Considerations

Specific software performs the resolution of the I2P Special-Use Domain Names presented in this document; this resolution process happens outside of the scope of DNS. Leakage of requests to such domains to the global operational DNS can cause interception of traffic that might be misused to monitor, censor, or abuse the user's trust, and lead to privacy issues with potentially tragic consequences for the user.

This document reserves these Top-Level Domain names to minimize the possibility of confusion, conflict, and especially privacy risks for users.

In the introduction of this document, there's a requirement that DNS operators do not override resolution of the I2P Names. This is a regulatory measure and cannot prevent such malicious abuse in practice. Its purpose is to limit any information leak that would result from incorrectly configured systems, and to avoid that resolvers make unnecessary contact to the DNS Root Zone for such domains. Verisign, Inc., as well as several Internet service providers (ISPs) have notoriously abused their position to override NXDOMAIN responses to their customers in the past [SSAC-NXDOMAIN-Abuse]. For example, if a DNS operator would decide to override NXDOMAIN and send advertising to leaked .onion sites, the information leak to the DNS would extend to the advertising server, with unpredictable consequences. Thus, implementors should be aware that any positive response coming from DNS must be considered with extra care, as it suggests a leak to DNS has been made, contrary to user's privacy expectations.

The reality of X.509 Certificate Authorities (CAs) creating misleading certificates for I2P pTLDs due to ignorance stresses the need to document their special use. Given the nature of "B32.I2P",

X.509 Certificate Authorities MAY create certificates for such domains given CSRs signed with the respective private keys corresponding to the respective names.

6. IANA Considerations

The Internet Assigned Numbers Authority (IANA) reserved the following entries in the Special-Use Domain Names registry [RFC6761]:

.i2p

[TO REMOVE: the assignement URL is https://www.iana.org/assignments/special-use-domain-names/]

7. Acknowledgements

The authors thank the I2P and Namecoin developers for their constructive feedback, as well as Mark Nottingham for his proof-reading and valuable feedback. The authors also thank the members of DNSOP WG for their critiques and suggestions.

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.

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

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

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

8.2. Informative References

[I2P-CHOICE] Hacker, J. and The I2P Community, "I2P Dev Meeting 059", September 2003, <https://geti2p.net/en/meetings/059>.

[I2P-NAMING]
Hacker, J. and The I2P Community, "Naming in I2P and Addressbook", April 2014, <https://geti2p.net/en/docs/naming>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[SAC45] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System", November 2010, <http://www.icann.org/en/groups/ssac/documents/sac-045-en.pdf>.

[SSAC-NXDOMAIN-Abuse]
ICANN Security and Stability Advisory Committee, "Redirection in the COM and NET Domains", July 2004, <http://www.icann.org/committees/security/ssac-report-09jul04.pdf>.

[zzz2009] The I2P Project and L. Schimmer, "Peer Profiling and Selection in the I2P Anonymous Network", January 2009, <https://geti2p.net/_static/pdf/I2P-PET-CON-2009.1.pdf>.

Authors' Addresses

Christian Grothoff
INRIA
Equipe Decentralisee
INRIA Rennes Bretagne Atlantique
263 avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes, Bretagne F-35042
FR

Email: christian@grothoff.org

Matthias Wachs
Technische Universitaet Muenchen
Free Secure Network Systems Group
Lehrstuhl fuer Netzarchitekturen und Netzdienste
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: wachs@net.in.tum.de

Hellekin O. Wolf (editor)
GNU consensus

Email: hellekin@gnu.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Leif Ryge
Tor Project Inc.

Email: leif@synthesize.us

dnsop
Internet-Draft
Obsoletes: 5966 (if approved)
Updates: 1035,1123 (if approved)
Intended status: Standards Track
Expires: July 18, 2016

J. Dickinson
S. Dickinson
Sinodun
R. Bellis
ISC
A. Mankin
D. Wessels
Verisign Labs
January 15, 2016

DNS Transport over TCP - Implementation Requirements
draft-ietf-dnsop-5966bis-06

Abstract

This document specifies the requirement for support of TCP as a transport protocol for DNS implementations and provides guidelines towards DNS-over-TCP performance on par with that of DNS-over-UDP. This document obsoletes RFC5966 and therefore updates RFC1035 and RFC1123.

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 July 18, 2016.

Copyright Notice

Table of Contents

1. Introduction

Most DNS [RFC1034] transactions take place over UDP [RFC0768]. TCP [RFC0793] is always used for full zone transfers (AXFR) and is often used for messages whose sizes exceed the DNS protocol's original 512-byte limit. The growing deployment of DNSSEC and IPv6 has increased response sizes and therefore the use of TCP. The need for increased TCP use has also been driven by the protection it provides against address spoofing and therefore exploitation of DNS in reflection/amplification attacks. It is now widely used in Response Rate Limiting [RRL1][RRL2]. Additionally, recent work on DNS privacy solutions such as [DNS-over-TLS] is another motivation to re-visit DNS-over-TCP requirements.

Section 6.1.3.2 of [RFC1123] states:

> DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries.

However, some implementors have taken the text quoted above to mean that TCP support is an optional feature of the DNS protocol.

The majority of DNS server operators already support TCP and the default configuration for most software implementations is to support TCP. The primary audience for this document is those implementors whose limited support for TCP restricts interoperability and hinders deployment of new DNS features.

This document therefore updates the core DNS protocol specifications such that support for TCP is henceforth a REQUIRED part of a full DNS protocol implementation.

There are several advantages and disadvantages to the increased use of TCP (see Appendix A) as well as implementation details that need to be considered. This document addresses these issues and presents TCP as a valid transport alternative for DNS. It extends the content of [RFC5966], with additional considerations and lessons learned from research, developments and implementation of TCP in DNS and in other internet protocols.

Whilst this document makes no specific requirements for operators of DNS servers to meet, it does offer some suggestions to operators to help ensure that support for TCP on their servers and network is optimal. It should be noted that failure to support TCP (or the blocking of DNS over TCP at the network layer) will probably result in resolution failure and/or application-level timeouts.

2. Requirements 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 [RFC2119].

3. Terminology

- Persistent connection: a TCP connection that is not closed either by the server after sending the first response nor by the client after receiving the first response.

- Connection Reuse: the sending of multiple queries and responses over a single TCP connection.

- Idle DNS-over-TCP session: Clients and servers view application level idleness differently. A DNS client considers an established DNS-over-TCP session to be idle when it has no pending queries to send and there are no outstanding responses. A DNS server considers an established DNS-over-TCP session to be idle when it has sent responses to all the queries it has received on that connection.

- Pipelining: the sending of multiple queries and responses over a single TCP connection but not waiting for any outstanding replies before sending another query.

- Out-Of-Order Processing: The processing of queries concurrently and the returning of individual responses as soon as they are available, possibly out-of-order. This will most likely occur in recursive servers, however it is possible in authoritative servers that, for example, have different backend data stores.

4. Discussion

In the absence of EDNS0 (Extension Mechanisms for DNS 0 [RFC6891]) (see below), the normal behaviour of any DNS server needing to send a UDP response that would exceed the 512-byte limit is for the server to truncate the response so that it fits within that limit and then set the TC flag in the response header. When the client receives such a response, it takes the TC flag as an indication that it should retry over TCP instead.

RFC 1123 also says:

> ... it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that

applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

Existing deployments of DNS Security (DNSSEC) [RFC4033] have shown that truncation at the 512-byte boundary is now commonplace. For example, a Non-Existent Domain (NXDOMAIN) (RCODE == 3) response from a DNSSEC-signed zone using NextSECure 3 (NSEC3) [RFC5155] is almost invariably larger than 512 bytes.

Since the original core specifications for DNS were written, the Extension Mechanisms for DNS have been introduced. These extensions can be used to indicate that the client is prepared to receive UDP responses larger than 512 bytes. An EDNS0-compatible server receiving a request from an EDNS0-compatible client may send UDP packets up to that client’s announced buffer size without truncation.

However, transport of UDP packets that exceed the size of the path MTU causes IP packet fragmentation, which has been found to be unreliable in many circumstances. Many firewalls routinely block fragmented IP packets, and some do not implement the algorithms necessary to reassemble fragmented packets. Worse still, some network devices deliberately refuse to handle DNS packets containing EDNS0 options. Other issues relating to UDP transport and packet size are discussed in [RFC5625].

The MTU most commonly found in the core of the Internet is around 1500 bytes, and even that limit is routinely exceeded by DNSSEC-signed responses.

The future that was anticipated in RFC 1123 has arrived, and the only standardised UDP-based mechanism that may have resolved the packet size issue has been found inadequate.

5. Transport Protocol Selection

Section 6.1.3.2 of [RFC1123] is updated: All general-purpose DNS implementations MUST support both UDP and TCP transport.

- Authoritative server implementations MUST support TCP so that they do not limit the size of responses to what fits in a single UDP packet.

- Recursive server (or forwarder) implementations MUST support TCP so that they do not prevent large responses from a TCP-capable server from reaching its TCP-capable clients.

- Stub resolver implementations (e.g., an operating system's DNS resolution library) MUST support TCP since to do otherwise would limit the interoperability between their own clients and upstream servers.

Regarding the choice of when to use UDP or TCP, Section 6.1.3.2 of RFC 1123 also says:

> ... a DNS resolver or server that is sending a non-zone-transfer query MUST send a UDP query first.

This requirement is hereby relaxed. Stub resolvers and recursive resolvers MAY elect to send either TCP or UDP queries depending on local operational reasons. TCP MAY be used before sending any UDP queries. If the resolver already has an open TCP connection to the server it SHOULD reuse this connection. In essence, TCP ought to be considered a valid alternative transport to UDP, not purely a retry option.

In addition it is noted that all Recursive and Authoritative servers MUST send responses using the same transport as the query arrived on. In the case of TCP this MUST also be the same connection.

6. Connection Handling

6.1. Current practices

Section 4.2.2 of [RFC1035] says:

- The server should assume that the client will initiate connection closing, and should delay closing its end of the connection until all outstanding client requests have been satisfied.

- If the server needs to close a dormant connection to reclaim resources, it should wait until the connection has been idle for a period on the order of two minutes. In particular, the server should allow the SOA and AXFR request sequence (which begins a refresh operation) to be made on a single connection. Since the server would be unable to answer queries anyway, a unilateral close or reset may be used instead of graceful close.

Other more modern protocols (e.g., HTTP/1.1 [RFC7230], HTTP/2 [RFC7540]) have support by default for persistent TCP connections for all requests. Connections are then normally closed via a 'connection close' signal from one party.

The description in [RFC1035] is clear that servers should view connections as persistent (particularly after receiving an SOA), but unfortunately does not provide enough detail for an unambiguous interpretation of client behaviour for queries other than a SOA. Additionally, DNS does not yet have a signalling mechanism for connection timeout or close, although some have been proposed.

6.1.1. Clients

There is no clear guidance today in any RFC as to when a DNS client should close a TCP connection, and there are no specific recommendations with regard to DNS client idle timeouts. However, at the time of writing, it is common practice for clients to close the TCP connection after sending a single request (apart from the SOA/ AXFR case).

6.1.2. Servers

Many DNS server implementations use a long fixed idle timeout and default to a small number of TCP connections. They also offer little by the way of TCP connection management options. The disadvantages of this include:

- Operational experience has shown that long server timeouts can easily cause resource exhaustion and poor response under heavy load.

- Intentionally opening many connections and leaving them idle can trivially create a TCP "denial-of-service" attack as many DNS servers are poorly equipped to defend against this by modifying their idle timeouts or other connection management policies.

- A modest number of clients that all concurrently attempt to use persistent connections with non-zero idle timeouts to such a server could unintentionally cause the same "denial-of-service" problem.

Note that this denial-of-service is only on the TCP service. However, in these cases it affects not only clients wishing to use TCP for their queries for operational reasons, but all clients who choose to fall back to TCP from UDP after receiving a TC=1 flag.

6.2. Recommendations

The following sections include recommendations that are intended to result in more consistent and scalable implementations of DNS-over-TCP.

6.2.1. Connection Re-use

One perceived disadvantage to DNS over TCP is the added connection setup latency, generally equal to one RTT. To amortize connection setup costs, both clients and servers SHOULD support connection reuse by sending multiple queries and responses over a single persistent TCP connection.

When sending multiple queries over a TCP connection clients MUST NOT re-use the DNS Message ID of an in-flight query on that connection in order to avoid Message ID collisions. This is especially important if the server could be performing out-of-order processing (see Section 7).

6.2.1.1. Query Pipelining

Due to the historical use of TCP primarily for zone transfer and truncated responses, no existing RFC discusses the idea of pipelining DNS queries over a TCP connection.

In order to achieve performance on par with UDP DNS clients SHOULD pipeline their queries. When a DNS client sends multiple queries to a server, it SHOULD NOT wait for an outstanding reply before sending the next query. Clients SHOULD treat TCP and UDP equivalently when considering the time at which to send a particular query.

It is likely that DNS servers need to process pipelined queries concurrently and also send out-of-order responses over TCP in order to provide the level of performance possible with UDP transport. If TCP performance is of importance, clients might find it useful to use server processing times as input to server and transport selection algorithms.

DNS servers (especially recursive) MUST expect to receive pipelined queries. The server SHOULD process TCP queries concurrently, just as it would for UDP. The server SHOULD answer all pipelined queries, even if they are received in quick succession. The handling of responses to pipelined queries is covered in Section 7.

6.2.2. Concurrent connections

To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimize the number of concurrent TCP connections made to any individual server. It is RECOMMENDED that for any given client/server interaction there SHOULD be no more than one connection for regular queries, one for zone transfers and one for each protocol that is being used on top of TCP, for example, if the resolver was using TLS. It is however noted that certain primary/secondary

configurations with many busy zones might need to use more than one TCP connection for zone transfers for operational reasons (for example, to support concurrent transfers of multiple zones).

Similarly, servers MAY impose limits on the number of concurrent TCP connections being handled for any particular client IP address or subnet. These limits SHOULD be much looser than the client guidelines above, because the server does not know, for example, if a client IP address belongs to a single client or is multiple resolvers on a single machine, or multiple clients behind a device performing Network Address Translation (NAT).

6.2.3. Idle Timeouts

To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimize the idle time of established DNS-over-TCP sessions made to any individual server. DNS clients SHOULD close the TCP connection of an idle session, unless an idle timeout has been established using some other signalling mechanism, for example, [edns-tcp-keepalive].

To mitigate the risk of unintentional server overload it is RECOMMENDED that the default server application-level idle period be of the order of seconds, but no particular value is specified. In practice, the idle period can vary dynamically, and servers MAY allow idle connections to remain open for longer periods as resources permit. A timeout of at least a few seconds is advisable for normal operations to support those clients that expect the SOA and AXFR request sequence to be made on a single connection as originally specified in [RFC1035]. Servers MAY use zero timeouts when experiencing heavy load or are under attack.

DNS messages delivered over TCP might arrive in multiple segments. A DNS server that resets its idle timeout after receiving a single segment might be vulnerable to a "slow read attack." For this reason, servers SHOULD reset the idle timeout on the receipt of a full DNS message, rather than on receipt of any part of a DNS message.

6.2.4. Tear Down

Under normal operation DNS clients typically initiate connection closing on idle connections, however DNS servers can close the connection if their local idle timeout policy is exceeded. Connections can be also closed by either end under unusual conditions such as defending against an attack or system failure/reboot.

DNS Clients SHOULD retry unanswered queries if the connection closes before receiving all outstanding responses. No specific retry algorithm is specified in this document.

If a DNS server finds that a DNS client has closed a TCP session, or if the session has been otherwise interrupted, before all pending responses have been sent then the server MUST NOT attempt to send those responses. Of course the DNS server MAY cache those responses.

7. Response Reordering

RFC 1035 is ambiguous on the question of whether TCP responses may be reordered -- the only relevant text is in Section 4.2.1, which relates to UDP:

> Queries or their responses may be reordered by the network, or by processing in name servers, so resolvers should not depend on them being returned in order.

For the avoidance of future doubt, this requirement is clarified. Authoritative servers and recursive resolvers are RECOMMENDED to support the preparing of responses in parallel and sending them out-of-order, regardless of the transport protocol in use. Stub and recursive resolvers MUST be able to process responses that arrive in a different order to that in which the requests were sent, regardless of the transport protocol in use.

In order to achieve performance on par with UDP, recursive resolvers SHOULD process TCP queries in parallel and return individual responses as soon as they are available, possibly out-of-order.

Since pipelined responses can arrive out-of-order, clients MUST match responses to outstanding queries on the same TCP connection using the Message ID. If the response contains a question section the client MUST match the QNAME, QCLASS and QTYPE fields. Failure by clients to properly match responses to outstanding queries can have serious consequences for interoperability.

8. TCP Message Length Field

DNS clients and servers SHOULD pass the two-octet length field, and the message described by that length field, to the TCP layer at the same time (e.g., in a single "write" system call) to make it more likely that all the data will be transmitted in a single TCP segment. This is both for reasons of efficiency and to avoid problems due to some DNS server implementations behaving undesirably when reading data from the TCP layer (due to a lack of clarity in previous

standards). For example, some DNS server implementations might abort a TCP session if the first "read" from the TCP layer does not contain both the length field and the entire message.

To clarify, DNS servers MUST NOT close a connection simply because the first "read" from the TCP layer does not contain the entire DNS message, and servers SHOULD apply the connection timeouts as specified in Section 6.2.3.

9. TCP Fast Open

This section is non-normative.

TCP Fast Open [RFC7413] (TFO) allows data to be carried in the SYN packet, reducing the cost of re-opening TCP connections. It also saves up to one RTT compared to standard TCP.

TFO mitigates the security vulnerabilities inherent in sending data in the SYN, especially on a system like DNS where amplification attacks are possible, by use of a server-supplied cookie. TFO clients request a server cookie in the initial SYN packet at the start of a new connection. The server returns a cookie in its SYN-ACK. The client caches the cookie and reuses it when opening subsequent connections to the same server.

The cookie is stored by the client’s TCP stack (kernel) and persists if either the client or server processes are restarted. TFO also falls back to a regular TCP handshake gracefully.

DNS services taking advantage of IP anycast [RFC4786] might need to take additional steps when enabling TFO. From [RFC7413]:

> Servers that accept connection requests to the same server IP address should use the same key such that they generate identical Fast Open Cookies for a particular client IP address. Otherwise a client may get different cookies across connections; its Fast Open attempts would fall back to regular 3WHS.

When DNS-over-TCP is a transport for DNS private exchange, as in [DNS-over-TLS], the implementor needs to be aware of TFO and to ensure that data requiring protection (e.g. data for a DNS query) is not accidentally transported in the clear. See [DNS-over-TLS] for discussion."

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

Some DNS server operators have expressed concern that wider promotion and use of DNS over TCP will expose them to a higher risk of denial-of-service (DoS) attacks on TCP (both accidental and deliberate).

Although there is a higher risk of some specific attacks against TCP-enabled servers, techniques for the mitigation of DoS attacks at the network level have improved substantially since DNS was first designed.

Readers are advised to familiarise themselves with [CPNI-TCP], a security assessment of TCP detailing known TCP attacks and countermeasures which references most of the relevant RFCs on this topic.

To mitigate the risk of DoS attacks, DNS servers are advised to engage in TCP connection management. This could include maintaining state on existing connections, re-using existing connections and controlling request queues to enable fair use. It is likely to be advantageous to provide configurable connection management options, for example:

- total number of TCP connections
- maximum TCP connections per source IP address or subnet
- TCP connection idle timeout
- maximum DNS transactions per TCP connection
- maximum TCP connection duration

No specific values are recommended for these parameters.

Operators are advised to familiarise themselves with the configuration and tuning parameters available in the operating system TCP stack. However detailed advice on this is outside the scope of this document.

Operators of recursive servers are advised to ensure that they only accept connections from expected clients (for example by the use of an ACL), and do not accept them from unknown sources. In the case of UDP traffic, this will help protect against reflection attacks

[RFC5358] and in the case of TCP traffic it will prevent an unknown client from exhausting the server's limits on the number of concurrent connections.

12. Acknowledgements

The authors would like to thank Francis Dupont and Paul Vixie for detailed review, Andrew Sullivan, Tony Finch, Stephane Bortzmeyer, Joe Abley, Tatuya Jinmei and the many others who contributed to the mailing list discussion. Also Liang Zhu, Zi Hu, and John Heidemann for extensive DNS-over-TCP discussions and code. Lucie Guiraud and Danny McPherson for reviewing early versions of this document. We would also like to thank all those who contributed to RFC5966.

13. References

13.1. Normative References

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

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[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>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[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>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <http://www.rfc-editor.org/info/rfc4786>.

[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>.

[RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <http://www.rfc-editor.org/info/rfc5358>.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <http://www.rfc-editor.org/info/rfc5625>.

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

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

13.2. Informative References

[Connection-Oriented-DNS] Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A., and N. Somaiya, "Connection-Oriented DNS to Improve Privacy and Security", <http://www.isi.edu/~johnh/PAPERS/Zhu15b.pdf>.

[CPNI-TCP]
CPNI, "Security Assessment of the Transmission Control Protocol (TCP)", 2009, <http://www.gont.com.ar/papers/tn-03-09-security-assessment-TCP.pdf>.

[DNS-over-TLS]
Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "TLS for DNS: Initiation and Performance Considerations", draft-ietf-dprive-dns-over-tls (work in progress), January 2016.

[edns-tcp-keepalive]
Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", draft-ietf-dnsop-edns-tcp-keepalive-05 (work in progress), Jan 2015.

[fragmentation-considered-poisonous]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", May 2012, <http://arxiv.org/abs/1205.4011>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RRL1] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1-Draft1, August 2014, <http://ss.vix.su/~vixie/isc-tn-2012-1.txt>.

[RRL2] "BIND RRL", ISC Knowledge Base AA-00994, April 2012, <https://deepthought.isc.org/article/AA-00994/0/Using-the-Response-Rate-Limiting-Feature-in-BIND-9.10.html>.

Appendix A. Summary of Advantages and Disadvantages to using TCP for DNS

The TCP handshake generally prevents address spoofing and, therefore, the reflection/amplification attacks which plague UDP.

IP fragmentation is less of a problem for TCP than it is for UDP. TCP stacks generally implement Path MTU Discovery so they can avoid IP fragmentation of TCP segments. UDP, on the other hand, does not provide reassembly, which means datagrams that exceed the path MTU size must experience fragmentation [RFC5405]. Middleboxes are known to block IP fragments, leading to timeouts and forcing client implementations to "hunt" for EDNS0 reply size values supported by the network path. Additionally, fragmentation may lead to cache poisoning [fragmentation-considered-poisonous].

TCP setup costs an additional RTT compared to UDP queries. Setup costs can be amortized by reusing connections, pipelining queries, and enabling TCP Fast Open.

TCP imposes additional state-keeping requirements on clients and servers. The use of TCP Fast Open reduces the cost of closing and re-opening TCP connections.

Long-lived TCP connections to anycast servers might be disrupted due to routing changes. Clients utilizing TCP for DNS need to always be prepared to re-establish connections or otherwise retry outstanding queries. It might also be possible for TCP Multipath [RFC6824] to allow a server to hand a connection over from the anycast address to a unicast address.

There are many "Middleboxes" in use today that interfere with TCP over port 53 [RFC5625]. This document does not propose any solutions, other than to make it absolutely clear that TCP is a valid transport for DNS and support for it is a requirement for all implementations.

A more in-depth discussion of connection orientated DNS can be found elsewhere [Connection-Oriented-DNS].

Appendix B. Changes between revisions

[Note to RFC Editor: please remove this section prior to publication.]

B.1. Changes -05 to -06

Introduction: Add reference to DNS-over-TLS

Section 5: ’s/it/the resolver/’ and ’s/fallback/retry/’

Section 6.1.1: Make clear behaviour is ’at the time of writing’, not a recommendation

Section 6.2.1.1: Change SHOULD to MUST.

Section 6.2.2: Clarify ’operational reasons’ for zone transfers.

Section 8: Re-word to remove reference to TCP segments.

Section 9: Add sentence about use of TFO with DNS privacy solutions.

B.2. Changes -04 to -05

Added second RRL reference to introduction

Introduction, paragraph 5: s/may result/will probably result/

Section 5: Strengthened wording on update of RFC1123

Section 5: Added reference to HTTP/2

Section 6.2.1: Simplify wording of Message ID collisions

Section 6.2.2: Clarify wording on idle timeout reset

Section 6.2.4: Use DNS Server/client for consistency

Section 8: Re-word to reduce confusion of timeout vs TCP reads

Appendix C: Updated differences to RFC5966.

B.3. Changes -03 to -04

- Re-stated how messages received over TCP should be mapped to queries.
- Added wording to cover timeouts for server side behaviour for when receiving TCP messages.
- Added sentence to abstract stating this obsoletes RFC5966.
- Moved reference to RFC6891 earlier in Discussion section.
- Several minor wording updates to improve clarity.
- Corrected nits and updated references.

B.4. Changes -02 to -03

- Replaced certain lower case RFC2119 keywords to improve clarity.
- Updated section 6.2.2 to recognise requirements for concurrent zone transfers.
- Changed ’client IP address’ to ’client IP address or subnet’ when discussing restrictions on TCP connections from clients.
- Added reference to edns-tcp-keepalive draft.
- Added wording to introduction to reference Appendix A and state TCP is a valid transport alternative for DNS.
- Improved description of CPNI-TCP as a general reference source on TCP security related RFCs.

B.5. Changes -01 to -02

- Added more text to Introduction as background to TCP use.
- Added definitions of Persistent connection and Idle session to Terminology section.
- Separated Connection Handling section into Current Practice and Recommendations. Provide more detail on current practices and divided Recommendations up into more granular sub-sections.
- Add section on Idle time with new text on recommendations for client idle behaviour.
- Move TCP message field length discussion to separate section.
- Removed references to system calls in TFO section.
- Added more discussion on DoS mitigation in Security Considerations section.
- Added statement that servers MAY use 0 idle timeout.
- Re-stated position of TCP as an alternative to UDP in Discussion.
- Updated text on server limits on concurrent connections from a particular client.
- Added text that client retry logic is outside the scope of this document.

- Clarified that servers should answer all pipelined queries even if sent very close together.

B.6. Changes -00 to -01

- Changed updates to obsoletes RFC 5966.
- Improved text in Section 4 Transport Protocol Selection to change "TCP SHOULD NOT be used only for the transfers and as a fallback" to make the intention clearer and more consistent.
- Reference to TCP FASTOPEN updated now that it is an RFC.
- Added paragraph to say that implementations MUST NOT send the TCP framing 2 byte length field in a separate packet to the DNS message.
- Added Terminology section.
- Changed should and RECOMMENDED in reference to parallel processing to SHOULD in sections 7 and 8.
- Added text to address what a server should do when a client closes the TCP connection before pending responses are sent.
- Moved the Advantages and Disadvantages section to an appendix.

Appendix C. Changes to RFC5966

[Note to RFC Editor: please leave this section in the final document.]

This document obsoletes [RFC5966] and differs from it in several respects. An overview of the most substantial changes/updates that implementors should take note of is given below:

1. A Terminology section (Section 3) is added defining several new concepts.
2. Paragraph 3 of Section 5 puts TCP on a more equal footing with UDP than RFC5966. For example it states:
    1. TCP MAY be used before sending any UDP queries.
    2. TCP ought to be considered a valid alternative transport to UDP, not purely a fallback option.

3. Section 6.2.1 adds a new recommendation that TCP connection-reuse SHOULD be supported.

4. Section 6.2.1.1 adds a new recommendation that DNS clients SHOULD pipeline their queries and DNS servers SHOULD process pipelined queries concurrently.

5. Section 6.2.2 adds new recommendations on the number and usage of TCP connections for client/server interactions.

6. Section 6.2.3 adds a new recommendation that DNS clients SHOULD close idle sessions unless using a signalling mechanism.

7. Section 7 clarifies that servers are RECOMMENDED to prepare TCP responses in parallel and send answers out-of-order. It also clarifies how TCP queries and responses should be matched by clients.

8. Section 8 adds a new recommendation about how DNS clients and servers should handle the 2 byte message length field for TCP messages.

9. Section 9 adds a non-normative discussion of the use of TCP Fast Open.

10. The Section 11 adds new advice regarding DoS mitigation techniques.

Authors' Addresses

John Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
UK

Email: jad@sinodun.com
URI: http://sinodun.com

Sara Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
UK

Email: sara@sinodun.com
URI: http://sinodun.com

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

Phone: +1 650 423 1200
Email: ray@isc.org
URI: http://www.isc.org

Allison Mankin
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
US

Phone: +1 703 948-3200
Email: amankin@verisign.com

Duane Wessels
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
US

Phone: +1 703 948-3200
Email: dwessels@verisign.com

dnsop
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2016

P. Wouters
Red Hat
J. Abley
Dyn, Inc.
S. Dickinson
Sinodun
R. Bellis
ISC

February 22, 2016

The edns-tcp-keepalive EDNS0 Option
draft-ietf-dnsop-edns-tcp-keepalive-06

Abstract

DNS messages between clients and servers may be received over either UDP or TCP. UDP transport involves keeping less state on a busy server, but can cause truncation and retries over TCP. Additionally, UDP can be exploited for reflection attacks. Using TCP would reduce retransmits and amplification. However, clients commonly use TCP only for retries and servers typically use idle timeouts on the order of seconds.

This document defines an EDNS0 option ("edns-tcp-keepalive") that allows DNS servers to signal a variable idle timeout. This signalling encourages the use of long-lived TCP connections by allowing the state associated with TCP transport to be managed effectively with minimal impact on the DNS transaction time.

Table of Contents

1. Introduction

DNS messages between clients and servers may be received over either UDP or TCP [RFC1035]. Historically, DNS clients used API's that only facilitated sending and receiving a single query over either UDP or TCP. New APIs and deployment of DNSSEC validating resolvers on hosts that in the past were using stub resolving only is increasing the DNS client base that prefer using long lived TCP connections. Long-lived TCP connections can result in lower request latency than the case where UDP transport is used and truncated responses are received. This is because clients that retry over TCP following a truncated UDP response typically only use the TCP session for a single (request, response) pair, continuing with UDP transport for subsequent queries.

The use of TCP transport requires state to be retained on DNS servers. If a server is to perform adequately with a significant query load received over TCP, it must manage its available resources to ensure that all established TCP sessions are well-used, and idle connections are closed after an appropriate amount of time.

UDP transport is stateless, and hence presents a much lower resource burden on a busy DNS server than TCP. An exchange of DNS messages over UDP can also be completed in a single round trip between communicating hosts, resulting in optimally-short transaction times. UDP transport is not without its risks, however.

A single-datagram exchange over UDP between two hosts can be exploited to enable a reflection attack on a third party. Response Rate Limiting [RRL] is designed to help mitigate such attacks against authoritative-only servers. One feature of RRL is to let some amount of responses "slip" through the rate limiter. These are returned with the TC (truncation) bit set, which causes legitimate clients to re-query using TCP transport.

[RFC1035] specified a maximum DNS message size over UDP transport of 512 bytes. Deployment of DNSSEC [RFC4033] and other protocols subsequently increased the observed frequency at which responses exceed this limit. EDNS0 [RFC6891] allows DNS messages larger than 512 bytes to be exchanged over UDP, with a corresponding increased incidence of fragmentation. Fragmentation is known to be problematic in general, and has also been implicated in increasing the risk of cache poisoning attacks [fragmentation-considered-poisonous].

TCP transport is less susceptible to the risks of fragmentation and reflection attacks. However, TCP transport for DNS as currently

deployed has expensive setup overhead, compared to using UDP (when no retry is required).

The overhead of the three-way TCP handshake for a single DNS transaction is substantial, increasing the transaction time for a single (request, response) pair of DNS messages from 1 x RTT to 2 x RTT. There is no such overhead for a session that is already established therefore the overhead of the initial TCP handshake is minimised when the resulting session is used to exchange multiple DNS message pairs over a single session. The extra RTT time for session setup can be represented as the equation (1 + N)/N, where N represents the number of DNS message pairs that utilize the session and the result approaches unity as N increases.

With increased deployment of DNSSEC and new RRtypes containing application specific cryptographic material, there is an increase in the prevalence of truncated responses received over UDP with retries over TCP. The overhead for a DNS transaction over UDP truncated due to RRL is 3x RTT, higher than the overhead imposed on the same transaction initiated over TCP.

This document proposes a signalling mechanism between DNS clients and servers that encourages the use of long-lived TCP connections by allowing the state associated with TCP transport to be managed effectively with minimal impact on the DNS transaction time.

This mechanism will be of benefit both for stub-resolver and resolver-authoritative TCP connections. In the latter case the persistent nature of the TCP connection can provide improved defence against attacks including DDoS.

The reduced overhead of this extension adds up significantly when combined with other EDNS0 extensions, such as [CHAIN-QUERY] and [DNS-over-TLS]. For example, the combination of these EDNS0 extensions make it possible for hosts on high-latency mobile networks to natively and efficiently perform DNSSEC validation and encrypt queries.

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].

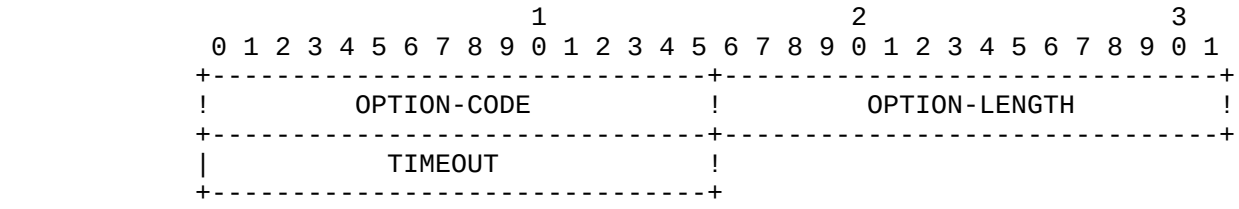
3. The edns-tcp-keepalive Option

This document specifies a new EDNS0 [RFC6891] option, edns-tcp-keepalive, which can be used by DNS clients and servers to signal a willingness to keep an idle TCP session open to conduct future DNS transactions, with the idle timeout being specified by the server. This specification does not distinguish between different types of DNS client and server in the use of this option.

[DRAFT-5966bis] defines an 'idle' DNS-over-TCP session from both the client and server perspective. The idle timeout described here begins when the idle condition is met per that definition and should be reset when that condition is lifted i.e. when a client sends a message or when a server receives a message on an idle connection.

3.1. Option Format

The edns-tcp-keepalive option is encoded as follows:



where:

OPTION-CODE: the EDNS0 option code assigned to edns-tcp-keepalive, TBD1

OPTION-LENGTH: the value 0 if the TIMEOUT is omitted, the value 2 if it is present;

TIMEOUT: an idle timeout value for the TCP connection, specified in units of 100 milliseconds, encoded in network byte order.

3.2. Use by DNS Clients

3.2.1. Sending Queries

DNS clients MUST NOT include the edns-tcp-keepalive option in queries sent using UDP transport.

DNS clients MAY include the edns-tcp-keepalive option in the first query sent to a server using TCP transport to signal their desire to keep the connection open when idle.

DNS clients MAY include the edns-tcp-keepalive option in subsequent queries sent to a server using TCP transport to signal their continued desire to keep the connection open when idle.

Clients MUST specify an OPTION-LENGTH of 0 and omit the TIMEOUT value.

3.2.2. Receiving Responses

A DNS client that receives a response using UDP transport that includes the edns-tcp-keepalive option MUST ignore the option.

A DNS client that receives a response using TCP transport that includes the edns-tcp-keepalive option MAY keep the existing TCP session open when it is idle. It SHOULD honour the timeout received in that response (overriding any previous timeout) and initiate close of the connection before the timeout expires.

A DNS client that receives a response that includes the edns-tcp-keepalive option with a TIMEOUT value of 0 SHOULD send no more queries on that connection and initiate closing the connection as soon as it has received all outstanding responses.

A DNS client that sent a query containing the edns-keepalive-option but receives a response that does not contain the edns-keepalive-option SHOULD assume the server does not support keepalive and behave following the guidance in [DRAFT-5966bis]. This holds true even if a previous edns-keepalive-option exchange occurred on the existing TCP connection.

3.3. Use by DNS Servers

3.3.1. Receiving Queries

A DNS server that receives a query using UDP transport that includes the edns-tcp-keepalive option MUST ignore the option.

A DNS server that receives a query using TCP transport that includes the edns-tcp-keepalive option MAY modify the local idle timeout associated with that TCP session if resources permit.

3.3.2. Sending Responses

A DNS server that receives a query sent using TCP transport that includes an OPT RR (with or without the edns-tcp-keepalive option) MAY include the edns-tcp-keepalive option in the response to signal the expected idle timeout on a connection. Servers MUST specify the TIMEOUT value that is currently associated with the TCP session. It

is reasonable for this value to change according to local resource constraints. The DNS server SHOULD send a edns-tcp-keepalive option with a timeout of 0 if it deems its local resources are too low to service more TCP keepalive sessions, or if it wants clients to close currently open connections.

3.4. TCP Session Management

Both DNS clients and servers are subject to resource constraints which will limit the extent to which TCP sessions can persist. Effective limits for the number of active sessions that can be maintained on individual clients and servers should be established, either as configuration options or by interrogation of process limits imposed by the operating system. Servers that implement edns-tcp-keepalive should also engage in TCP connection management by recycling existing connections when appropriate, closing connections gracefully and managing request queues to enable fair use.

In the event that there is greater demand for TCP sessions than can be accommodated, servers may reduce the TIMEOUT value signalled in successive DNS messages to minimise idle time on existing sessions. This also allows, for example, clients with other candidate servers to query to establish new TCP sessions with different servers in expectation that an existing session is likely to be closed, or to fall back to UDP.

Based on TCP session resources servers may signal a TIMEOUT value of 0 to request clients to close connections as soon as possible. This is useful when server resources become very low or a denial-of-service attack is detected and further maximises the shifting of TIME_WAIT state to well-behaved clients.

However it should be noted that RCF6891 states:

> Lack of presence of an OPT record in a request MUST be taken as an indication that the requestor does not implement any part of this specification and that the responder MUST NOT include an OPT record in its response.

Since servers must be faithful to this specification even on a persistent TCP connection it means that (following the initial exchange of timeouts) a server may not be presented with the opportunity to signal a change in the idle timeout associated with a connection if the client does not send any further requests containing EDNS0 OPT RRs. This limitation makes persistent connection handling via an initial idle timeout signal more attractive than a mechanism that establishes default persistence and

then uses a connection close signal (in a similar manner to HTTP 1.1 [RFC7320]).

If a client includes the edns-tcp-keepalive option in the first query, it SHOULD include an EDNS0 OPT RR periodically in any further messages it sends during the TCP session. This will increase the chance of the client being notified should the server modify the timeout associated with a session. The algorithm for choosing when to do this is out of scope of this document and is left up to the implementor and/or operator.

DNS clients and servers MAY close a TCP session at any time in order to manage local resource constraints. The algorithm by which clients and servers rank active TCP sessions in order to determine which to close is not specified in this document.

3.5. Non-Clean Paths

Many paths between DNS clients and servers suffer from poor hygiene, limiting the free flow of DNS messages that include particular EDNS0 options, or messages that exceed a particular size. A fallback strategy similar to that described in [RFC6891] section 6.2.2 SHOULD be employed to avoid persistent interference due to non-clean paths.

3.6. Anycast Considerations

DNS servers of various types are commonly deployed using anycast [RFC4786].

Changes in network topology between clients and anycast servers may cause disruption to TCP sessions making use of edns-tcp-keepalive more often than with TCP sessions that omit it, since the TCP sessions are expected to be longer-lived. It might be possible for anycast servers to avoid disruption due to topology changes by making use of TCP multipath [RFC6824] to anchor the server side of the TCP connection to an unambiguously-unicast address.

4. Intermediary Considerations

It is RECOMMENDED that DNS intermediaries which terminate TCP connections implement edns-tcp-keepalive. An intermediary that does not implement edns-tcp-keepalive but sits between a client and server that both support edns-tcp-keepalive might close idle connections unnecessarily.

5. Security Considerations

The edns-tcp-keepalive option can potentially be abused to request large numbers of long-lived sessions in a quick burst. When a DNS Server detects abusive behaviour, it SHOULD immediately close the TCP connection and free the resources used.

Servers could choose to monitor client behaviour with respect to the edns-tcp-keepalive option to build up profiles of clients that do not honour the specified timeout.

Readers are advised to familiarise themselves with the security considerations outlined in [DRAFT-5966bis]

6. IANA Considerations

The IANA is directed to assign an EDNS0 option code for the edns-tcp-keepalive option from the DNS EDNS0 Option Codes (OPT) registry as follows:

Value	Name	Status	Reference
TBD1	edns-tcp-keepalive	Standard	[This document]

7. Acknowledgements

The authors acknowledge the contributions of Jinmei TATUYA and Mark Andrews. Thanks to Duane Wessels for detailed review and the many others who contributed to the mailing list discussion.

8. References

8.1. Normative References

[DRAFT-5966bis] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", draft-ietf-dnsop-5966bis (work in progress), January 2016.

[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>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <http://www.rfc-editor.org/info/rfc4786>.

[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>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

8.2. Informative References

[CHAIN-QUERY] Wouters, P., "Chain Query requests in DNS", draft-ietf-dnsop-edns-chain-query (work in progress), January 2016.

[DNS-over-TLS] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "TLS for DNS: Initiation and Performance Considerations", draft-ietf-dprive-dns-over-tls (work in progress), January 2016.

[fragmentation-considered-poisonous] Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", arXiv 1205.4011, May 2012, <http://arxiv.org/abs/1205.4011>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1-Draft1, April 2012, <http://ss.vix.su/~vixie/isc-tn-2012-1.txt>.

Appendix A. Editors' Notes

A.1. Abridged Change History

[Note to RFC Editor: please remove this section prior to publication.]

A.1.1. draft-ietf-dnsop-edns-tcp-keepalive-06

Introduction: Moved paragraph 8 to paragraph 2 for readability.

Introduction: clarified that TCP has expensive setup overhead compared to UDP.

Section 3: Add explicit description of the idle timeout.

Section 3.3.2, 1st para: make explicit that query may or may not contain edns-tcp-keepalive option.

Section 3.3.2: remove discussion of intermediary behaviour.

A.1.2. draft-ietf-dnsop-edns-tcp-keepalive-05

Reword Abstract and paragraph 9 in Introduction to remove discussion on balancing UDP/TCP and talk about encouraging use of long-lived TCP sessions.

Section 3.2.2: should -> SHOULD

Changed draft-ietf-dnsop-5966bis to be a normative reference, therefore adding a dependancy on publication of that as RFC.

Reword sentence referring to RFC6824 since it is informational.

Update IANA option to Standard.

Remove last sentence from 1st paragraph of introduction.

Reword paragraph 6 in Introduction, merge paragraph 7 and 8.

Reword Section 3, first sentence to clarify the timeout is specified by the server.

Correct missing URIs in 2 references.

Clarify statement in Section 3.2.2 as how clients should handle updating the timeout when receiving a response.

Reworded first paragraph of Introduction discussing TCP vs (UDP + retry over TCP). Changed 'fallback' to 'retry' in 2 places.

A.1.3. draft-ietf-dnsop-edns-tcp-keepalive-04

Adding wording to sections 3.2.1 and 3.4 to clarify client behaviour on subsequent queries on a TCP connection.

Changed the should to a SHOULD in section 3.2.2

Changed Nameserver to DNS server in section 5.

Updated references.

Changed reference to RFC6824 to be informative.

Corrected reference to requested EDNS0 option code to be 'TBD1'.

A.1.4. draft-ietf-dnsop-edns-tcp-keepalive-03

Clarified that a response to a query with any OPT RR may contain the ends-tcp-keepalive option.

Corrected TIMEOUT length from 4 to 2 in the diagram.

Updated references, including name change of STARTTLS -> DNS-over-TLS and adding reference for cache poisoning.

Updated wording in section on Intermediary Considerations.

Updated wording describing RRL.

Added paragraph to security section describing client behaviour profiles.

Added wording to introduction on use case for stub/resolver/ authoritative.

A.1.5. draft-ietf-dnsop-edns-tcp-keepalive-02

Changed timeout value to idle timeout and re-phrased document around this.

Changed units of timeout to 100ms to allow values less than 1 second.

Change specification to remove use of the option over UDP. This is potentially confusing, could cause issues with ALG's and adds only limited value.

Changed semantics so the client no longer sends a timeout. The client timeout is of limited value as servers should be managing connections based on their view of their resources, not on client requests as this is open to abuse. Additionally this identifies cases were the option is simply being reflected back.

Changed semantics for the meaning of a server sending a timeout of 0. The maximum timeout value of 6553.5s (~1.8h) is already large and a distinct 'connection close'-like signal is potentially more useful.

Added more detail on server side requirements when supporting keepalive in terms of resource and connection management.

Added discussion of EDNS0 per-message limitation and implications of this.

Added reference to STARTTLS draft and RFC7320.

A.1.6. draft-ietf-dnsop-edns-tcp-keepalive-01

Version bump with no changes

A.1.7. draft-ietf-dnsop-edns-tcp-keepalive-00

Clarifications, working group adoption.

A.1.8. draft-wouters-edns-tcp-keepalive-01

Also allow clients to specify KEEPALIVE timeout values, clarify motivation of document.

A.1.9. draft-wouters-edns-tcp-keepalive-00

Initial draft.

Authors' Addresses

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

Sara Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
UK

Email: sara@sinodun.com
URI: http://sinodun.com

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

Phone: +1 650 423 1200
Email: ray@isc.org
URI: http://www.isc.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 19, 2017

J. Ableyy
Dyn, Inc.
J. Schlyter
Kirei AB
G. Bailey
Independent
P. Hoffman
ICANN

July 18, 2016

DNSSEC Trust Anchor Publication for the Root Zone
draft-jabley-dnssec-trust-anchor-16

Abstract

The root zone of the Domain Name System (DNS) has been cryptographically signed using DNS Security Extensions (DNSSEC).

In order to obtain secure answers from the root zone of the DNS using DNSSEC, a client must configure a suitable trust anchor. This document describes the format and publication mechanisms IANA has used to distribute the DNSSEC trust anchors.

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 January 19, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) is described in [RFC1034] and [RFC1035]. Security extensions to the DNS (DNSSEC) are described in [RFC4033], [RFC4034], [RFC4035], [RFC4509], [RFC5155] and [RFC5702].

A discussion of operational practices relating to DNSSEC can be found in [RFC6781].

In the DNSSEC protocol, resource record sets (RRSets) are signed cryptographically. This means that a response to a query contains signatures that allow the integrity and authenticity of the RRSet to be verified. DNSSEC signatures are validated by following a chain of signatures to a "trust anchor". The reason for trusting a trust

anchor is outside the DNSSEC protocol, but having one or more trust anchors is required for the DNSSEC protocol to work.

The publication of trust anchors for the root zone of the DNS is an IANA function performed by ICANN. A detailed description of corresponding key management practices can be found in [DPS], which can be retrieved from the IANA Repository at <https://www.iana.org/dnssec/>.

This document describes the formats and distribution methods of DNSSEC trust anchors that have been used by IANA for the root zone of the DNS since 2010. Other organizations might have different formats and mechansims for distributing DNSSEC trust anchors for the root zone; however, most operators and software vendors have chosen to rely on the IANA trust anchors.

IMPORTANT NOTE: at the time of this writing, IANA intends to change the formats and distribution methods in the future. If such a change happens, IANA will publish the changes on its web site at <https://www.iana.org/dnssec/files>.

The formats and distribution methods described in this document are a complement to, not a substitute for, the automated DNSSEC trust anchor update protocol described in [RFC5011]. That protocol allows for secure in-band succession of trust anchors when trust has already been established. This document describes one way to establish an initial trust anchor that can be used by [RFC5011].

1.1. Definitions

The term "trust anchor" is used in many different contexts in the security community. Many of the common definitions conflict because they are specific to a specific system, such as just for DNSSEC or just for S/MIME messages.

In cryptographic systems with hierarchical structure, a trust anchor is an authoritative entity for which trust is assumed and not derived. The format of the entity differs in different systems, but the basic idea, that trust is assumed and not derived, is common to all the common uses of the term "trust anchor".

The root zone trust anchor formats published by IANA are defined in Section 2. [RFC4033] defines a trust anchor as "A configured DNSKEY RR or DS RR hash of a DNSKEY RR". Note that the formats defined here do not match the definition of "trust anchor" from [RFC4033]; however, a system that wants to convert the trusted material from IANA into a DS RR can do so.

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. IANA DNSSEC Root Zone Trust Anchor Formats and Semantics

IANA publishes trust anchors for the root zone in three formats:

- an XML document that contains the hashes of the DNSKEY records
- certificates in PKIX format [RFC5280] that contain DS records and the full public key of DNSKEY records
- certificate signing requests (CSRs) in PKCS#10 format [RFC2986] that contain DS records and the full public key of DNSKEY records

These formats and the semantics associated with each are described in the rest of this section.

2.1. Hashes in XML

The XML document contains a set of hashes for the DNSKEY records that can be used to validate the root zone. The hashes are consistent with the defined presentation format of Delegation Signer (DS) resource records from [RFC4034].

2.1.1. XML Syntax

A Relax NG Compact Schema for the documents used to publish trust anchors is given in Figure 1.

```
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = element TrustAnchor {
    attribute id { xsd:string },
    attribute source { xsd:string },
    element Zone { xsd:string },

    keydigest+
}

keydigest = element KeyDigest {
    attribute id { xsd:string },
    attribute validFrom { xsd:dateTime },
    attribute validUntil { xsd:dateTime }?,

    element KeyTag {
            xsd:nonNegativeInteger { maxInclusive = "65535" } },
    element Algorithm {
            xsd:nonNegativeInteger { maxInclusive = "255" } },
    element DigestType {
            xsd:nonNegativeInteger { maxInclusive = "255" } },
    element Digest { xsd:hexBinary }
}
```

Figure 1

2.1.2. XML Semantics

The TrustAnchor element is the container for all of the trust anchors in the file.

The id attribute in the TrustAnchor element is an opaque string that identifies the set of trust anchors. Its value has no particular semantics. Note that the id element in the TrustAnchor element is different than the id element in the KeyDigest element, described below.

The source attribute in the TrustAnchor element gives information about where to obtain the TrustAnchor container. It is likely to be a URL, and is advisory only.

The Zone element in the TrustAnchor element states to which DNS zone this container applies. The root zone is indicated by a single period (.) character, without any quotation marks.

The TrustAnchor element contains one or more KeyDigest elements. Each KeyDigest element represents the digest of a DNSKEY record in the zone defined in the Zone element.

The id attribute in the KeyDigest element is an opaque string that identifies the hash. Its value is used in the file names and URI of the other trust anchor formats. This is described in Section 3.1. For example, if the value of the id attribute in the KeyDigest element is "Kjqmt7v", the URI for the CSR that is associated with this hash will be <https://data.iana.org/root-anchors/Kjqmt7v.csr>. Note that the id element in the KeyDigest element is different than the id element in the TrustAnchor element, described above.

The validFrom and validUntil attributes in the KeyDigest element specify the range of times that the KeyDigest element can be used as a trust anchor. Note that the KeyDigest element is optional; if it is not given, the trust anchor can be used until a KeyDigest element covering the same DNSKEY record but having a validUntil attribute is trusted by the relying party. Relying parties SHOULD NOT use a KeyDigest outside of the time range given in the validFrom and validUntil attributes.

The KeyTag element in the KeyDigest element contains the key tag for the DNSKEY record represented in this KeyDigest.

The Algorithm element in the KeyDigest element contains the signing algorithm identifier for the DNSKEY record represented in this KeyDigest.

The DigestType element in the KeyDigest element contains the digest algorithm identifier for the DNSKEY record represented in this KeyDigest.

The Digest element in the KeyDigest element contains the hexadecimal representation of the hash for the DNSKEY record represented in this KeyDigest.

2.1.3. Converting from XML to DS Records

The display format for the DS record that is the equivalent of a KeyDigest element can be constructed by marshaling the KeyTag, Algorithm, DigestType, and Digest elements. For example, assume that the TrustAnchor element contains:

```
<?xml version="1.0" encoding="UTF-8"?>
<TrustAnchor
   id="AD42165F-3B1A-4778-8F42-D34A1D41FD93"
   source="http://data.iana.org/root-anchors/root-anchors.xml">
<Zone>.</Zone>
<KeyDigest id="Kjqmt7v" validFrom="2010-07-15T00:00:00+00:00">
<KeyTag>19036</KeyTag>
<Algorithm>8</Algorithm>
<DigestType>2</DigestType>
<Digest>
49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
</Digest>
</KeyDigest>
</TrustAnchor>
```

The DS record would be:

```
. IN DS 19036 8 2
   49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
```

2.1.4. XML Example

Figure 2 describes two fictitious trust anchors for the root zone.

```
<?xml version="1.0" encoding="UTF-8"?>

<TrustAnchor
    id="AD42165F-B099-4778-8F42-D34A1D41FD93"
    source="http://data.iana.org/root-anchors/root-anchors.xml">
    <Zone>.</Zone>
    <KeyDigest id="42"
               validFrom="2010-07-01T00:00:00-00:00"
               validUntil="2010-08-01T00:00:00-00:00">
        <KeyTag>34291</KeyTag>
        <Algorithm>5</Algorithm>
        <DigestType>1</DigestType>
        <Digest>c8cb3d7fe518835490af8029c23efbce6b6ef3e2</Digest>
    </KeyDigest>
    <KeyDigest id="53"
               validFrom="2010-08-01T00:00:00-00:00">
        <KeyTag>12345</KeyTag>
        <Algorithm>5</Algorithm>
        <DigestType>1</DigestType>
        <Digest>a3cf809dbdbc835716ba22bdc370d2efa50f21c7</Digest>
    </KeyDigest>
</TrustAnchor>
```

Figure 2

2.2. Certificates

Each public key that can be used as a trust anchor is represented as a certificate in PKIX format. Each certificate is signed by the ICANN certificate authority. The SubjectPublicKeyInfo in the certificate represents the public key of the KSK. The Subject field has the following attributes:

O: the string "ICANN".

OU: the string "IANA".

CN: the string "Root Zone KSK" followed by the time and date of key generation in the format specified in [RFC3339]. For example, a CN might be "Root Zone KSK 2010-06-16T21:19:24+00:00".

resourceRecord: a string in the presentation format of the Delegation Signer (DS) [RFC4034] resource record for the DNSSEC public key.

The "resourceRecord" attribute in the Subject is defined as follows:

```
ResourceRecord
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-dns-resource-record(70) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS

caseIgnoreMatch FROM SelectedAttributeTypes
    { joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 4 }

;

iana OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
    dod(6) internet(1) private(4) enterprise(1) 1000 }

iana-dns OBJECT IDENTIFIER ::= { iana 53 }

resourceRecord ATTRIBUTE ::= {
    WITH SYNTAX IA5String
    EQUALITY MATCHING RULE caseIgnoreMatch
    ID iana-dns
}

END
```

2.3. Certificate Signing Requests

Each public key that can be used as a trust anchor is represented as a certificate signing request (CSR) in PKCS#10 format. The SubjectPublicKeyInfo and Subject field are the same as for certificates (see Section 2.2 above).

3. Root Zone Trust Anchor Retrieval

3.1. Retrieving Trust Anchors with HTTPS and HTTP

Trust anchors are available for retrieval using HTTPS and HTTP.

In this section, all URLs are given using the "https:" scheme. If HTTPS cannot be used, replace the "https:" scheme with "http:".

The URL for retrieving the set of hashes described in Section 2.1 is <https://data.iana.org/root-anchors/root-anchors.xml>.

The URL for retrieving the PKIX certificate described in Section 2.2 is <https://data.iana.org/root-anchors/KEYDIGEST-ID.crt>, with the string "KEYDIGEST-ID" replaced the "id" attribute from the KeyDigest element from the XML file, as described in Section 2.1.2.

The URL for retrieving the CSR described in Section 2.3 is <https://data.iana.org/root-anchors/KEYDIGEST-ID.csr>, with the string "KEYDIGEST-ID" replaced the "id" attribute from the KeyDigest element from the XML file, as described in Section 2.1.2.

4. Accepting DNSSEC Trust Anchors

A validator operator can choose whether or not to accept the trust anchors described in this document using whatever policy they want. In order to help validator operators verify the content and origin of trust anchors they receive, IANA uses digital signatures that chain to an ICANN-controlled CA over the trust anchor data.

It is important to note that the ICANN CA is not a DNSSEC trust anchor. Instead, it is an optional mechanism for verifying the content and origin of the XML and certificate trust anchors. It is also important to note that the ICANN CA cannot be used to verify the origin of the trust anchor in the CSR format.

The content and origin of the XML file can be verified using a digital signature on the file. IANA provides a detached CMS [RFC5652] signature that chains to the ICANN CA with the XML file. The URL for a detached CMS signature for the XML file is <https://data.iana.org/root-anchors/root-anchors.p7s>.

(IANA also provided a detached OpenPGP [RFC4880] signature as a second parallel verification mechanism for the first trust anchor publication, but has indicated that it will not use this parallel mechanism in the future.)

Another method IANA uses to help validator operators verify the content and origin of trust anchors they receive is to use the TLS protocol for distributing the trust anchors. Currently, the CA used for data.iana.org is well-known, that is, one that is a WebTrust-accredited Certificate Authority. If a system retrieving the trust anchors trusts the CA that IANA uses for the "data.iana.org" web server, HTTPS SHOULD be used instead of HTTP in order to have assurance of data origin.

5. IANA Considerations

This document defines id-mod-dns-resource-record, value 70 (see Section 2.2), in the SMI Security for PKIX Module Identifier registry.

Beyond the IANA registry action above, this document makes no other requests and places no further restrictions on IANA.

6. Security Considerations

This document describes how DNSSEC trust anchors for the root zone of the DNS are published. Many DNSSEC clients will only configure IANA-issued trust anchors for the DNS root to perform validation. As a consequence, reliable publication of trust anchors is important.

This document aims to specify carefully the means by which such trust anchors are published, as an aid to the formats and retrieval methods described here being integrated usefully into user environments.

7. Acknowledgements

Many pioneers paved the way for the deployment of DNSSEC in the root zone of the DNS, and the authors hereby acknowledge their substantial collective contribution.

This document incorporates suggestions made by Alfred Hoenes and Russ Housley, whose contributions are appreciated.

8. References

8.1. 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>.

[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>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <http://www.rfc-editor.org/info/rfc2986>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[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>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc4035>.

[RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, DOI 10.17487/RFC4509, May 2006, <http://www.rfc-editor.org/info/rfc4509>.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <http://www.rfc-editor.org/info/rfc5011>.

[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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, DOI 10.17487/RFC5702, October 2009, <http://www.rfc-editor.org/info/rfc5702>.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <http://www.rfc-editor.org/info/rfc6781>.

8.2. Informative References

[DPS] Ljunggren, F., Okubo, T., Lamb, R., and J. Schlyter, "DNSSEC Practice Statement for the Root Zone KSK Operator", October 2010, <https://www.iana.org/dnssec/icann-dps.txt>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

Appendix A. Historical Note

The first KSK for use in the root zone of the DNS was generated at a key ceremony at an ICANN Key Management Facility (KMF) in Culpeper, Virginia, USA on 2010-06-16. This key entered production during a second key ceremony held at an ICANN KMF in El Segundo, California, USA on 2010-07-12. The resulting trust anchor was first published on 2010-07-15.

Appendix B. About this Document

[RFC Editor: please remove this section, including all subsections, prior to publication.]

B.1. Discussion

This document is not the product of any IETF working group. However, communities interested in similar technical work can be found at the IETF in the DNSOP and DNSEXT working groups.

The team responsible for deployment of DNSSEC in the root zone can be reached at rootsign@icann.org.

The authors also welcome feedback sent to them directly.

Authors' Addresses

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

Jakob Schlyter
Kirei AB

Email: jakob@kirei.se

Guillaume Bailey
Independent

Email: GuillaumeBailey@outlook.com

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 7, 2016

W. Kumari
Google
G. Huston
APNIC
E. Hunt
Internet Systems Consortium
R. Arends
ICANN

October 5, 2015

Signalling of DNS Security (DNSSEC) Trust Anchors
draft-wkumari-dnsop-trust-management-01

Abstract

[Editor note: This originally included a mechanism to actually roll the keys (like RFC5011 does), but feedback from the Prague meeting indicated a strong preference for signalling only.]

This document describes a simple method for validating recursive resolvers to signal their configured list of DNSSEC trust anchors. This mechanism allows the trust anchor maintainer to monitor the progress of the migration to a new trust anchor, and so predict the effect before decommissioning the existing trust anchor.

It is primarily aimed at the root DNSSEC trust anchor, but should be applicable to trust anchors elsewhere in the DNS as well.

[Ed note - informal summary: One of the big issues with rolling the root key is that it is unclear who all is using RFC5011, who all has successfully fetched and installed the new key, and, most importantly, who all will die when the old key is revoked. By having resolvers query for a special QNAME, comprised of the list of TAs that it knows about, we effectively signal "up stream" to the authoritative server. By querying for this name, the recursive exposes its list of TAs to this authoritative server. This allows the TA maintainer to gather information relating to the state of TAs on resolvers.]

[Ed note: Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication.]

[This document is being collaborated on in Github at: https://github.com/wkumari/draft-wkumari-dnsop-trust-management. The most recent version of the document, open issues, etc should all be available here. The authors (gratefully) accept pull requests]

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When a DNSSEC-aware resolver performs validation, it requires a trust anchor to validate the DNSSEC chain. An example of a trust anchor is the so called DNSSEC "root key". For a variety of reasons, this trust anchor may need to be replaced or "rolled", to a new key (potentially with a different algorithm, different key length, etc.).

[RFC5011] provides a secure mechanism to do this, but operational experience has demonstrated a need for some additional functionality that was not foreseen.

During the current efforts to roll the IANA DNSSEC "root key", it has become clear that, in order to predict (and minimize) outages caused by rolling the key, real-time information about the uptake of the new key will be needed.

This document defines a mechanism ("trust anchor telemetry") by which validating resolvers can provide information about their configured trust anchors. Readers of this document are expected to be familiar with the contents of [RFC7344] and [RFC5011].

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. Trust Anchor Telemetry

The purpose of the mechanism described in this document is to allow the trust anchor maintainer to determine how widely deployed a given trust anchor is. This information is signaled from the validating resolver to the authoritative servers serving the zone in which the trust anchor lives by sending a periodic query to that zone. The query type of the TAT Query is NULL. The query name is a TAT Name, a format which encodes the list of the trust anchors for that zone that are currently in use by the validating resolver, along with status information about each key. Telemetry information can be retrieved by the trust anchor maintainer by examining logged queries that match the TAT Name format.

2.1. TAT Name Format

The TAT Name is generated as follows:

1. For each trust anchor that the resolver knows and/or is using, generate a string consisting of the key's Algorithm in decimal format, followed by an underscore ('_'), followed by the derived Key Tag in decimal format. [NOTE: If we used hex, this could just be AAKKKK, no need for a punctuation mark, but it would be less human-readable.]

2. Follow each string with a character indicating the status of the key from the resolver's point of view:

   S Static trust anchor, not subject to [RFC5011]

   A Accepted trust anchor

   P Pending trust anchor, not yet accepted

   R Revoked trust anchor

3. Sort the list in numerically ascending order of Algorithm and Key Tag.

4. Concatenate the list, with each string used as a label in a domain name.

5. Append _tat.<domain>

Assuming no more than two digits for the Algorithm and five for the Key Tag, a TAT Name for the root zone can encode up to 24 trust anchors. [Someone should probably check my math. QUESTION: Do we need to specify what will happen in the crazy case of a resolver having configured more than 24 trust anchors? -each]

Examples:

- If the resolver has a single trust anchor statically configured for the root zone, with an algorithm of RSASHA256 and a Key Tag of 19036, it would emit a query for 8_19036S._tat.

- If the resolver were configured to use [RFC5011] trust anchor management, it would send 8_19036A._tat.

- If a new key with Key Tag 1999 was added to the root zone and had been seen by the resolver, but was too recent to have been accepted as a trust anchor, then the resolver would send a query

for 8_1999P.8_19036A._tat. After the hold-down timer ([RFC5011] Section 2.2) had expired, the resolver would send a query for 8_1999A.8_19036A._tat.

- o If there is a separate static trust anchor configured for example.com with an algorithm of RSASHA1 and a Key Tag of 1234, the resolver would send a query for 5_1234S._tat.example.com.

NOTE: The format of the TAT Name requires that Key Tags MUST be unique, at least within "recent" history. If (e.g. during a Key Ceremony) a new DNSKEY is generated whose derived Key Tag collides with an exiting one (statistically unlikely, but not impossible) this DNSKEY MUST NOT be used, and a new DNSKEY MUST be generated. [Ed note: This is to prevent two successive keys having the same keytag (e.g: 123), and then seeing "8_123A." - which 123 key was that?! RFC4034 Appendix B admonition: "Implementations MUST NOT assume that the key tag uniquely identifies a DNSKEY RR", but this appears to be targeted at validating resolver implmentations.]

3. Sending the Trust Anchor Telemetry Query

When a compliant validating resolver performs the "Active Refresh" query as part of its RFC5011 ([RFC5011] Section 2.3)) processing it will also send a query for the TAT Name. This SHOULD be the default for compliant resolvers.

It will receive back either a negative response (e.g. NXDOMAIN), or a (nonsensical) answer. As the entire purpose of this query is to send information from a recursive resolver to the nameservers that serve the zone containing a trust anchor, the response to the query contains no useful information and MUST be ignored.

4. Known issues and limitations

This solution is designed to provide a rough idea of the rate of uptake of a new key during a key rollover; perfect visibility is not achievable. In particular:

1. Only compliant resolvers will send telemetry queries; no information is provided from legacy resolvers, or from those who choose to disable this functionality.

2. The trust anchor maintainer has no way to differentiate a query that is emitted by the resolver itself from a query that is forwarded through the resolver. (Note, however, that forwarded queries are likely to be infrequent; responses to TAT queries will in most cases be negatively cached with an NXDOMAIN covering

the _TAT subdomain; subsequent client queries would be answered from the cache rather than forwarded to the trust anchor zone.)

3. An attacker could forge TAT queries to trick the trust anchor maintainer into a false impression of the adoption rate of a new trust anchor, if there were a perceived advantage to doing so.

[Open Questions:

1: In order to disambiguate queries from resolvers versus those forwarded through resolvers (or being recursed because of users behind the resolver) we *could* add craziness like having resolvers include ephemeral UUIDs or something...). Is this worth doing? (Personally I think not...)

2: We *could* also specify that compliant resolvers MUST NOT forward queries of type TDS to try limit this. Worth doing? This is some of the reason for having a defined type.

3: The authorative server *could* return a record with a long TTL to stop queries (if it knows that it is not doing a rollover in the near future). This seems like a simple option, worth doing? (I think so). (each thinks not.)

5. IANA Considerations

[Ed note: This is largely a place holder. The real IANA considerations section will require updating things like the DPS, etc.]

The format of the TAT query requires that Key Tags MUST be unique, at least within an interval. If, during a Key Ceremony, a new DNSKEY is generated whose derived Key Tag collides with an exiting one (statistically unlikely, but not impossible) this DNSKEY MUST NOT be used, and a new DNSKEY MUST be generated.

There will need to be some text added to the DNSSEC Ceremony to handle this.

6. Security Considerations

[Ed note: a placeholder as well]

This mechanism causes a recursive resolver to disclose the list of trust anchors that it knows about to the authorative servers serving the zone containing the TA (or attackers able to monitor the path between these devices). It is conceviable that an attacker may be able to use this to determine that a resolver trusts an outdated /

revoked trust anchor and perform a MitM attack. This would also require the attacker to have factored the private key. This seems farfetched....

7. Contributors

A number of people contributed significantly to this document, including Joe Abley, Paul Wouters, Paul Hoffman. Wes Hardaker and David Conrad.

8. Acknowledgements

9. References

9.1. 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>.

[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, <http://www.rfc-editor.org/info/rfc4034>.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <http://www.rfc-editor.org/info/rfc5011>.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <http://www.rfc-editor.org/info/rfc7344>.

9.2. Informative References

[I-D.ietf-sidr-iana-objects] Manderson, T., Vegoda, L., and S. Kent, "RPKI Objects issued by IANA", draft-ietf-sidr-iana-objects-03 (work in progress), May 2011.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -00 to -01.1:

- Ripped all the actual keyroll logic out.
- Added Geoff, Evan and Roy as authors.
- Added some limitations and known issues.
- Renamed to TAT, added tag describing the state of the TA.

From -00.1 to -00 (published):

- Integrated comments and feedback from DRC and Paul Hoffman.
- Use _ as a prefix to make clear it is meta-type (drc)

From -00.0 to -00.1

- Initial draft, written in an airport lounge.

Authors' Addresses

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

Email: warren@kumari.net

Geoff Huston
APNIC
6 Cordelia St
South Brisbane QLD 4001
AUS

Email: gih@apnic.net

Evan Hunt
Internet Systems Consortium
950 Charter St
Redwood City, CA 94063
US

Email: each@isc.org

Roy Arends
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles, CA 90094-2536
US

Email: roy.arends@icann.org