

Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2021

R. Arends
M. Larson
ICANN
October 28, 2020

DNS Error Reporting
draft-arends-dns-error-reporting-00

Abstract

DNS Error Reporting is a lightweight error reporting mechanism that provides the operator of an authoritative server with reports on DNS resource records that fail to resolve or validate, that a Domain Owner or DNS Hosting organization can use to improve domain hosting. The reports are based on Extended DNS Errors [RFC8914].

When a domain name fails to resolve or validate due to a misconfiguration or an attack, the operator of the authoritative server may be unaware of this. To mitigate this lack of feedback, this document describes a method for a validating recursive resolver to automatically signal an error to an agent specified by the authoritative server. DNS Error Reporting uses the DNS to report errors.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 1, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction
2.	Requirements Notation
3.	Terminology
4.	Overview
4.1.	Managing Caching Optimizations
4.2.	Example
5.	EDNS0 Option Specification
6.	DNS Error Reporting Specification
6.1.	Reporting Resolver Specification
6.1.1.	Constructing the Reporting Query
6.2.	Authoritative Server Specification
6.3.	Reporting Agent Specification
6.4.	Choosing a Reporting Agent Domain
7.	Limitations
8.	IANA Considerations
9.	Security Considerations
10.	Acknowledgements
11.	Informative References
	Authors' Addresses

1. Introduction

When an authoritative server serves a stale DNSSEC signed zone, the cryptographic signatures over the resource record sets (RRsets) may have lapsed. A validating recursive resolver will fail to validate these resource records.

Similarly, when there is a mismatch between the DS records at a parent zone and the key signing key at the child zone, a validating recursive resolver will fail to authenticate records in the child zone.

These are two of several failure scenarios that may go unnoticed for some time by the operator of a zone.

There is no direct relationship between operators of validating recursive resolvers and authoritative servers. Outages are often noticed indirectly, by end users, and reported via social media, if reported at all.

When records fail to validate there is no facility to report this failure in an automated way. If there is any indication that an error or warning has happened, it is buried in log files of the validating resolver, if these errors are logged at all.

This document describes a facility that can be used by validating recursive resolvers to report errors in an automated way.

It allows an authoritative server to signal a reporting agent where the validating recursive resolver can report issues if it is configured to do so.

The burden of reporting a failure falls on the validating recursive resolver. It is important that the effort needed to report failure is low, with minimal impact to its main functions. To accomplish this goal, the DNS itself is utilized to report the error.

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. Terminology

Reporting Resolver: In the context of this document, the term reporting resolver is used as a shorthand for a validating recursive resolver that supports DNS Error Reporting.

Reporting Query: The DNS query used to report an error is called a reporting query. A reporting query is for DNS resource record type NULL. The details of the error report are encoded in the QNAME of the reporting query.

Reporting Agent: A facility responsible for receiving error reports on behalf of authoritative servers. This facility is indicated by a domain name.

Reporting Agent Domain: a domain name which the reporting resolver includes in the QNAME of the reporting query.

4. Overview

In a query-response exchange, a reporting resolver indicates support for DNS Error Reporting by including an EDNS option with OPTION-CODE [TBD] [RFC Editor: change TBD to the proper code when assigned by IANA.] and OPTION-LENGTH zero. The REPORTING AGENT DOMAIN field in the EDNS option is absent in a query.

An authoritative server indicates support for DNS Error Reporting by including an EDNS0 option with OPTION-CODE [TBD] [RFC Editor: change TBD to the proper code when assigned by IANA.] and the REPORTING AGENT DOMAIN in the option's payload. The authoritative server MUST NOT include this option if the reporting resolver has not signalled support for DNS Error Reporting. The authoritative server MUST NOT include this option in the response if the configured reporting agent domain is empty or the null label (the root).

When a reporting resolver sends a reporting query to report an error, it MUST NOT include the EDNS0 Error Reporting option in the reporting query. This avoids additional compounding error reporting when the reporting agent server is misconfigured.

To report an error, the reporting resolver encodes the error report in the QNAME of the reporting query. The reporting resolver builds this QNAME by concatenating the extended error code [RFC8914], the QTYPE and QNAME that resulted in failure, the label "_er", and the reporting agent domain. See the example in section 4.2. Note that a regular RCODE is not included, as the RCODE is not relevant to the extended error code.

The resulting concatenated domain name is sent as a standard DNS query for DNS resource record type NULL by the reporting resolver. This query MUST NOT have the EDNS0 option code [TBD] set to avoid compounding error notifications.

The query will ultimately arrive at an authoritative server of the reporting agent. A NODATA negative response is returned by the authoritative server of the reporting agent domain, which in turn can be cached by the reporting resolver.

This caching is essential. It ensures that the number of reports sent by a reporting resolver for the same problem is dampened, i.e.

once per TTL, however, certain optimizations such as [RFC8020] and [RFC8198] may reduce the error reporting.

4.1. Managing Caching Optimizations

The reporting resolver may utilize various caching optimizations that inhibit subsequent error reporting by the reporting resolver to the authoritative server for an agent domain.

If the authoritative server for the agent domain were to respond with NXDOMAIN (name error), [RFC8020] rules state that any name at or below that domain should be considered unreachable, and negative caching would prohibit subsequent queries for anything at or below that domain for a period of time, depending on the negative TTL [RFC2308].

Since the authoritative server for an agent domain may not know the contents of all the zones it acts as an agent for, it is crucial that the authoritative does not respond with NXDOMAIN, as that may inhibit subsequent queries. The use of a wildcard domain name [RFC4592] in the zone for the agent domain will ensure the RCODE is consistently NOERROR.

Considering the Resource Record type for this wildcard record, type NULL is prohibited in master zone files [RFC1035]. However, any type that is not special according to [RFC4592] section 4 will do, such as a TXT record with an email address for the reporting agent in the RDATA.

Wildcard expansion occurs, even if the QTYPE is not for the type owned by the wildcard domain name. The response is a "no error, but no data" response ([RFC4592], section 2.2.1.) that contains a NOERROR RCODE and empty answer section. Note that reporting resolvers are not expected to query for this TXT record, since reporting queries use type NULL. This record is solely present to ensure a NODATA response is returned in response to reporting queries.

When the zone for the reporting agent domain is signed, a resolver may utilize aggressive negative caching, discussed in [RFC8198]. This optimization makes use of NSEC and NSEC3 (without opt-out) records and allows the resolver to do the wildcard synthesis. When this happens, the resolver may not send subsequent queries as it will be able to synthesize a response from previously cached material.

A solution is to avoid DNSSEC for the reporting agent domain's zone. Signing the agent domain's zone will incur an additional burden on the reporting resolver, as it has to validate the response. However, this response has no utility to the reporting resolver.

If an operator does sign a reporting agent domain's zone for whatever reason, one option is to use NSEC3 with opt-out, as that configuration precludes wildcard synthesis on the resolver.

4.2. Example

The domain broken.test is hosted on a set of authoritative servers. One of these serves a stale version. This authoritative server has a reporting agent configured: a01.reporting-agent.example.

The reporting resolver is unable to validate the broken.test RRSset for type A, due to an RRSIG record with an expired signature.

The reporting resolver constructs the QNAME 7.1.broken.test._er.a01.reporting-agent.example and resolves it. This QNAME indicates extended DNS error 7 occurred while trying to validate broken.test type 1 (A) record.

After this query is received at one of the authoritative servers for the reporting agent domain (a01.reporting-agent.example), the reporting agent (the operators of the authoritative server for a01.reporting-agent.example) determines that the authoritative server for the broken.test zone suffers from an expired signature record (extended error 7) for type A for the domain name broken.test. The reporting agent can contact the operators of broken.test to fix the issue.

5. EDNS0 Option Specification

This method uses an EDNS0 [RFC6891] option to indicate support for sending DNS error reports and responding with the Reporting Agent Domain in DNS messages. The option is structured as follows:

```

      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          OPTION-CODE = TBD          |          OPTION-LENGTH          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                                REPORTING AGENT DOMAIN                                /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Field definition details:

- o OPTION-CODE, 2-octets/16-bits (defined in [RFC6891]), for indicating error reporting support is TBD. [RFC Editor: change TBD to the proper code when assigned by IANA.]
- o OPTION-LENGTH, 2-octets/16-bits ((defined in [RFC6891]) contains the length of the REPORTING AGENT DOMAIN field in octets.
- o REPORTING AGENT DOMAIN, a Domain name [RFC8499].

6. DNS Error Reporting Specification

The various errors that a reporting resolver may encounter are listed in [RFC8914]. Note that not all listed errors may be supported by the reporting resolver. This document does not specify what is an error and what is not.

The DNS class is not specified in the error report.

6.1. Reporting Resolver Specification

Reporting Resolvers may have a configuration that allows the following:

- o DNS Error Reporting level: warning and / or errors
- o Do nothing: the reporting resolver does not indicate support for DNS Error Reporting.
- o Report to Reporting Agent: Indicate DNS Error Reporting in queries and use the reporting agent specified in the EDNS0 option received from the authoritative server.

- o Report to Configured Agent: Use the reporting agent specified in local configuration. This may override or supplement "Reporting Agent Domain". The use for such an option could be to allow a recursive resolver to report all errors to a reporting agent of its choosing, not just in zones with DNS Error Reporting enabled.

The reporting resolver MUST NOT use DNS error reporting to report a failure in resolving the reporting query.

The reporting resolver MUST NOT use DNS error reporting if the authoritative server has an empty Reporting Agent Domain field in the EDNS Error Reporting option.

6.1.1. Constructing the Reporting Query

The QNAME for the reporting query is constructed by concatenating the following elements, appending each successive element in the list to the right-hand side of the QNAME:

- o The Extended DNS error, presented as a decimal value, in a single DNS label.
- o The QTYPE that was used in the query that resulted in the extended DNS error, presented as a decimal value, in a single DNS label.
- o The QNAME that was used in the query that resulted in the extended DNS error. The QNAME may consist of multiple labels and is concatenated as-is.
- o A label containing the string "_er".
- o The reporting agent domain. The reporting agent domain consists of multiple labels and is concatenated exactly as received in the EDNS option sent by the authoritative server.

If the resulting reporting query QNAME would exceed 255 octets, it MUST NOT be sent.

The purpose of the "_er" label is twofold. First, it allows the reporting agent to quickly differentiate between the agent domain and the faulty query name. Second, if the specified agent domain is empty, or a NULL label (even if it is not allowed in this specification), the reporting query will have "_er" as a top-level domain as a result and not the original query.

6.2. Authoritative Server Specification

The Authoritative Server MUST NOT have multiple reporting agent domains configured for a single zone. To support multiple reporting agents, a single agent can act as a syndicate to subsequently inform additional agents.

An authoritative server for a zone with DNS error reporting enabled MUST NOT also be authoritative for that zone's reporting agent domain's zone.

6.3. Reporting Agent Specification

While there are many zone configurations possible for the reporting agent domain, such as DNAME, CNAME or special delegation structures to redistribute errors, please note that the burden of reporting is on the reporting resolvers and that creating complicated

configurations that cause additional work for the reporting resolver on behalf of misconfigured servers is NOT RECOMMENDED.

It is RECOMMENDED that the reporting agent zone uses a wildcard DNS record of type TXT with an arbitrary string in the RDATA and a TTL of at least one hour.

6.4. Choosing a Reporting Agent Domain

Each authoritative server SHOULD be configured with a unique reporting agent domain. When different authoritative servers share the same reporting agent domain, it is not possible to determine which authoritative server the reported error relates to.

It is RECOMMENDED that the reporting agent domain be kept relatively short to allow for a longer QNAME in the reporting query.

While it may be obvious to use the hostname of the authoritative server as the reporting agent domain, it is not a requirement, as long as the reporting agent is able to map the reporting agent domain to the proper authoritative server. Using the hostname of the authoritative server as the reporting agent domain is NOT RECOMMENDED when the hostname has multiple addresses, or when addresses are anycast.

7. Limitations

The length of the owner name for which errors can be reported is limited due to the requirement to append the reporting agent domain and prepend the Extended Error value and the QTYPE to the reporting query's QNAME.

8. IANA Considerations

IANA is requested to assign the following DNS EDNS0 option code registry:

Value	Name	Status	Reference
-----	-----	-----	-----
TBD	DNS ERROR REPORT	Standard	[this document]

[RFC Editor: change TBD to the proper code when assigned by IANA.]

IANA is requested to assign the following Underscored and Globally Scoped DNS Node Name registry:

RR Type	_NODE NAME	Reference
-----	-----	-----
TXT	_er	[this document]

9. Security Considerations

Use of DNS Error Reporting may expose local configuration mistakes in the reporting resolver, such as stale DNSSEC trust anchors to the reporting agent.

DNS Error reporting SHOULD be done using DNS Query Name Minimization [RFC7816] to improve privacy.

DNS Error Reporting is done without any authentication between the reporting resolver and the authoritative server of the agent domain. Authentication significantly increases the burden on the reporting

resolver without any benefit to the reporting agent, authoritative server or reporting resolver.

The reporting resolver MUST NOT report about queries and responses from an encrypted channel (such as DNS over TLS [RFC7858] and DNS over HTTPS [RFC8484]).

The reporting resolver MUST NOT report about responses that did not match the qname/qtype/qclass and query-id in the original query [RFC5452], section 4.2.

The method described in this document will cause additional queries by the reporting resolver to authoritative servers in order to resolve the reporting query. This additional load is equivalent to the additional load when a resolver resolves the canonical name in a CNAME record.

This method can be abused by deploying broken zones with agent domains that are delegated to servers operated by the intended victim in combination with open resolvers [RFC8499]. This method MUST NOT be deployed by default on reporting resolvers and authoritative servers without requiring an explicit configuration element.

10. Acknowledgements

This document is based on an idea by Roy Arends and David Conrad.

11. Informative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<https://www.rfc-editor.org/info/rfc2308>>.
- [RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <<https://www.rfc-editor.org/info/rfc4592>>.
- [RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <<https://www.rfc-editor.org/info/rfc5452>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <<https://www.rfc-editor.org/info/rfc7816>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport

Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.

- [RFC8020] Bortzmeyer, S. and S. Huque, "NXDOMAIN: There Really Is Nothing Underneath", RFC 8020, DOI 10.17487/RFC8020, November 2016, <<https://www.rfc-editor.org/info/rfc8020>>.
- [RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <<https://www.rfc-editor.org/info/rfc8198>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <<https://www.rfc-editor.org/info/rfc8914>>.

Authors' Addresses

Roy Arends
ICANN

Email: roy.arends@icann.org

Matt Larson
ICANN

Email: matt.larson@icann.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 3 January 2022

J. Arkko
J. Novotny
Ericsson
2 July 2021

Privacy Improvements for DNS Resolution with Confidential Computing
draft-arkko-dns-confidential-02

Abstract

Data leaks are a serious privacy problem for Internet users. Data in flight and at rest can be protected with traditional communications security and data encryption. Protecting data in use is more difficult. In addition, failure to protect data in use can lead to disclosing session or encryption keys needed for protecting data in flight or at rest.

This document discusses the use of Confidential Computing, to reduce the risk of leaks from data in use. Our example use case is in the context of DNS resolution services. The document looks at the operational implications of running services in a way that even the owner of the service or compute platform cannot access user-specific information produced by the resolution process.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Background	4
3. Terminology	5
4. Prerequisites	5
5. Confidential Computing	6
6. Using Confidential Computing for DNS Resolution	7
7. Operational Considerations	9
7.1. Operations	9
7.2. Debugging	11
7.3. Dependencies	11
7.4. Additional services	12
7.5. Performance	12
8. Security Considerations	13
8.1. Observations from outside the TEE	13
8.2. Trust Relationships	13
8.3. Denial-of-Service Attacks	14
8.4. Other vulnerabilities	15
9. Recommendations	16
10. Acknowledgments	17
11. References	17
11.1. Normative References	17
11.2. Informative References	17
Authors' Addresses	22

1. Introduction

DNS privacy has been a popular topic in the last few years, and continues to be. The issues with regards to privacy are first that domain name meta-data is visible on the wire, even when the actual communications are encrypted. This is being addressed with better technology.

But even if the meta-data is hidden inside communications, any DNS resolvers still have the potential too see users' entire browsing history. This is particularly problematic, given that commonly used large public or operator resolver services are an obviously

attractive target, for both attacks and for commercial or other use of information visible to them.

A lot of work is ongoing in the industry and the IETF to address some of these issues:

- * Work on encrypted DNS query protocols to hide the meta-data related to domain names.
- * Discovery mechanisms. These may enable a bigger fraction of DNS query traffic to move to encrypted protocols, and may also help distributed queries to different parties to avoid concentrating all information in one place.
- * Practices, expectations, contracts (e.g., [RFC8932], Mozilla's trusted recursive resolver requirements [MozTRR])
- * Improvements outside DNS (e.g., encrypted Server Name Indication (eSNI) [I-D.ietf-tls-esni]).
- * General technology developments (e.g., confidential computing, attestations, remote attestation work at the IETF RATS WG, and so on)

The goal of this document is to build on all that work - and assume all communications are or become encrypted, including the DNS traffic. Our question is what problems remain? Is there a next step?

Our worry is that resolvers can be a major remaining source of leaks, e.g., through accidents, attacks, commercial use, or requests from the authorities. We need to protect user's data in flight, at rest, or in use - we wanted to experiment with technology that could reduce leaks on the last two cases. Confidential Computing is one such potential technology, but it is important to talk about it and get broader feedback. The use of this technology does have some operational impacts.

Our primary conclusions are that data held by servers should receive at least as much security attention as communications do. The authors feel that this is particularly crucial for DNS, due to the potential to leak of users' browsing histories, but principles apply also to other services.

As a result, all applicable tools should be considered, including confidential computing that is discussed in this document. However, the operational and business implications of such tools should be considered. Feedback to us is very welcome. Are these approaches

feasible or infeasible? What aspects need to be taken into account to successfully apply them?

2. Background

Communications security has been at the center of many security improvements in the Internet. The goal has been to ensure that communications are protected against outside observers and attackers [RFC3552] [RFC7258]. Communications security is, however, not sufficient by itself, and continuing success in better protection of communications is highlighting the need to address other issues.

In particular, more attention needs to be paid to protecting data not just in flight but also at rest or in use. User data leaks that can occur from servers and other systems, through accidents, attacks, commercial use of data, and requests for information by authorities. Both data at rest and data in use needs to be protected. Being able to protect data in use provides also benefits to protecting keys used for protecting data in flight and at rest.

Data leaks are very common, and include highly publicized ones or ones with significant consequences, such as [Cambridge]. Data leaks are also not limited to traditional computer applications, but can also impact anything from private health data [Vastaamo] to children's toys [Toys] or smart TVs [SmartTV].

The general issue and possible solutions have been discussed extensively elsewhere, e.g., [Digging], [Mem], [Comparison], [Innovative], [AMD], [Efficient], [CCC-Deepdive], [CC], and so on. The Internet-relevant angle has also been discussed in few documents, e.g., [I-D.lazanski-smart-users-internet], [I-D.iab-dedr-report] [I-D.arkko-farrell-arch-model-t-redux], and so on. The topic is also related to best practices for protocol and network architecture design, and what information can be provided to what participants in a system, see, e.g. [RFC8558] [I-D.thomson-tmi] [I-D.arkko-arch-infrastructure-centralisation].

Data leaks can occur in user-visible services that user has chosen to use and agreed to provide information to (at least in theory [Unread]). But leaks can also occur in other types of services, that are part of the infrastructure, such as DNS resolution services or parts of the communication infrastructure.

This document looks at the possibility of using a specific technical solution, Confidential Computing [CCC-Deepdive], to reduce the risk of leaks from data in use. We consider the operational implications of running services in a way that even the owner of the service or

compute platform cannot access user-specific information that is produced as a side-effect of the service.

We explore the use of Confidential Computing in the context of DNS resolution services [RFC1035]. This is a nice and relatively simple example, but there are of course potential other applications as well.

DNS resolution services are of course also an important case where privacy matters a lot for the users. Threats against the resolution process could prevent the user from accessing services. Data leaks from the process have the potential to expose the user's entire browsing history.

The use of Confidential Computing in the DNS context has been also discussed in other documents, e.g., [PDoT] and [I-D.reddy-add-server-policy-selection].

The DNS privacy issues have been also discussed in multiple documents, such as [RFC7626] [RFC8324] and so on.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Prerequisites

The primary sources of leaks are as follows:

- * Communications interception. This threat can be addressed by encrypted communications, such as the use of DNS-over-TLS (DoT) [RFC7858], DNS-over-HTTPS (DoH) [RFC8484], or DNS-over-QUIC (DoQ) [I-D.ietf-dprive-dnsquic] instead of traditional DNS protocols.
- * Data leakage from the server or service, either from data at rest or in use. This can be addressed by encrypting the data while at rest and employing the techniques discussed in this document for data in use.

The specific information that is privacy sensitive depends on the application. In DNS resolution application it is clear that the users' browsing histories, i.e., which users asked for what names is privacy sensitive, and protecting that information is the primary focus in this document. In contrast, the domains themselves or the

associated address information is in the general case public and not privacy sensitive. However, in some cases even this information may be sensitive, such as in the case of internal domains of a corporate network. Information not related to individuals may also be sensitive in some cases, e.g., the collective browsing destinations of an entire organization.

The above was also observed in [RFC7626] which stated the following:

"DNS data and the results of a DNS query are public [...], and may not have any confidentiality requirements. However, the same is not true of a single transaction or a sequence of transactions; that transaction is not / should not be public."

Nevertheless, it should be noted that technology can help only insofar as there is commercial willingness to provide the best possible service and to protect the users' information.

Similarly, the techniques discussed in this document are not the sole, or full answer to all problems. There are a lot of technical, operational, and governance issues that also matter and practices that help. A good compilation of some best practices can be found in [RFC8932], and particularly Section 5.2 that discusses data at rest.

5. Confidential Computing

Confidential Computing is about protecting data in use by performing computation in a hardware enforced Trusted Execution Environment (TEE) [CCC-Deepdive]. It addresses the need to protect data in use, which traditionally has been hard to achieve. It may also help improve the encryption of data in flight and at rest, by helping protect session keys and other security information used in that process.

For our purposes, we focus on Trusted Execution Environments that use computer hardware to provide the following characteristics:

- * **Attestability:** The environment can provide verifiable evidence to others (such as client using services running on it) about the environment, its characteristics, and the software it runs.
- * **Code integrity:** Unauthorized entities cannot modify software being run within the environment.
- * **Data confidentiality and integrity:** Unauthorized entities cannot view or modify data while it is in use within the TEE.

These characteristics have been paraphrased from [CCC-Deepdive]. See also [I-D.ietf-rats-architecture] for details of attestation. There are additional characteristics that matter in some situations, but for our purposes the above ones are central.

Specific technologies to perform Confidential Computing or run TEEs are becoming common in CPUs, operating systems, and other supporting software. For instance, Intel's Software Guard Extension (SGX) [SGX] is one CPU manufacturer's approach to this technology. SGX allows application developers to run software protected in a secure enclave protected by the CPU, including for instance encrypting all memory accesses outside the CPU and being able to provide remote attestation to outsiders about which software image is being run. These secure enclaves are the SGX approach to providing a TEE.

Confidential Computing is also becoming available on commonly available cloud computing services. When a user employs these services, they have the ability to run software and process data that even the owner of the cloud system does not have access to.

Interestingly, that is quite a contrast to the worries expressed some years ago about Trusted Computing technology, when it was feared that it enabled running software in users' computers that could act against the interests of the user in some cases, such as when protecting media files [Stallman]. While those concerns may apply even today in some cases, it is clear that whe the user can get secure information about services running somewhere in the network, this is an advantage for the users.

Note that availability might be another desirable characteristic for Confidential Computing systems, but it is one that is not in any special way supported by current technology. Ultimately, the owner of the computer still has the ability to choose when to switch the computer off, for instance. There is also no particular hardware technology at this time to deal with Denial-of-Service attacks. Some of the software techniques related to dealing with Denial-of-Service attacks are discussed in the Security Considerations section.

6. Using Confidential Computing for DNS Resolution

Confidential Computing can be used to provide a privacy-friendly resolution service in a server.

The basic arrangement is two-fold:

- * User's computer and the DNS resolution server communicate using an encrypted and integrity protected transport protocol, such as DoT or DoH [RFC7858] [RFC8484].

- * The secure connection terminates inside a TEE running in the the DNS resolution server. This TEE performs all the necessary processing to respond to the user's query. The TEE will not provide any user-specific information outside of the TEE, such as logs of what names specific clients queried for.

The TEE may need to contact other local servers or in the Internet to resolve a query that has no recently cached answer. We will discuss later how this can be done securely: it is necessary to prevent the linking any external actions such as receiving a client request and observing a query going out to other DNS servers in the Internet.

The arrangement is shown in Figure 1.

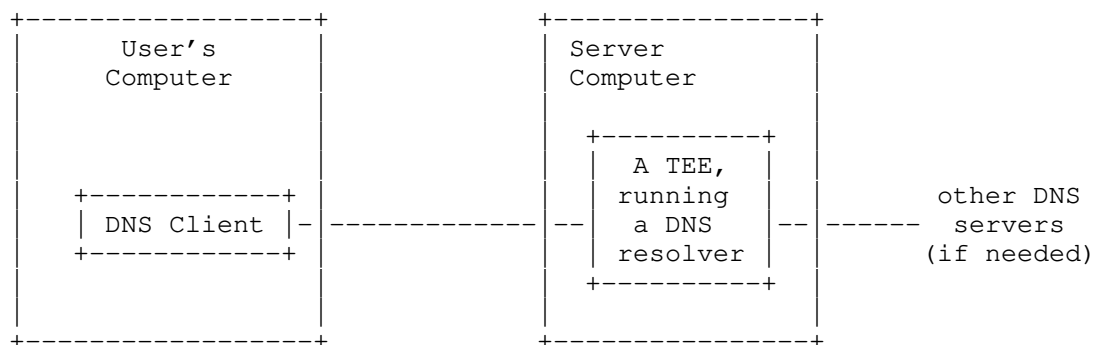


Figure 1: Confidential Computing for DNS Resolution

In this application, we strive to have no data at rest at all, at least nothing that relates directly to users. Data in flight and data in use are both protected by encryption. As a result of running the resolution service in this manner, any user-specific information should remain within the TEE, and not exposed to outsiders or even the owner of the service or the compute platform where the service is running in.

The authors believe that this is a desirable property. However, it remains to assure users and clients that the service is actually run in this manner. This can be done in two ways:

- * Through off-line reliance on a particular service, i.e., a human decision to use a particular system. Once there is a decision to use a particular system, cryptographic means such as public keys may be used to ensure that the client is indeed connected to the expected server. However, there is no guarantee that the human-

space statements about the practices used in running the server are valid.

- * Cryptographic check that the service is actually running inside a valid TEE and that it runs the expected software. Such checks needs to rely on third parties. The attestation verification is performed by a verifier - that can be either user's computer or a designated verifier as discussed in [I-D.ietf-rats-architecture] and [I-D.voit-rats-attestation-results] The verifier checks that (a) the cryptographic attestation refers to a server machine that is acceptable to the user (e.g., manufactured by a manufacturer it trusts, CPU features considered secure are used, features considered insecure are turned off, etc.) (b) that the software image designated as being run in the attestation is a software image that the relying party (end user) is willing to use (e.g., has a hash that matches a known software that does not log user actions, or is vouched as trustworthy by another party that the relying party trusts).

7. Operational Considerations

This section discusses some aspects of the Confidential Computing arrangement for DNS, based on the authors' experience with these systems.

7.1. Operations

Given that the service executes confidentially, and is not observable even by the owner of the hardware, the operations model becomes different. Some different models may be applied:

- * The service executes on a hardware platform (such as a commercial cloud service) that has no access to information, but there is some other management entity that does have access. The control functions of this entity can communicate with the service instances running in TEEs, and have access to the internal state and statistics of the service instances.
- * Truly confidential operations where the service and hardware owners have decided to deploy a service that really does not expose private user information to anyone, including themselves.

It is not clear how the first model differs from currently deployed service models. It merely makes it possible to run a service without exposing information to, say, the cloud provider, but any data collection about user behaviours would still be possible for the service owner.

As a result, this document focuses mostly on the second model. For some functions, such as DNS resolution, it is possible to hide all user-related information, and our document argues that we should do so.

Of course, the owners of a service do need some information to run the service, from an efficiency, scaling, problem tracking, and security monitoring point of view. The service operator may even benefit from seeing some overall trend information about various queries and traffic. This does not have to mean exposing individual user behaviours, however.

The authors have worked with aggregate statistics to be able to provide load, performance, memory usage, cache statistics, error, and other information out of the confidential processes. This helps the operator understand the health and status of various service instances. Even with aggregate statistics, there are some danger of revealing private information. For instance, even a sum of counters across all clients can reveal counters associated with an individual user, if the aggregate counters can be sampled at any time with arbitrary precision. For instance, the actions of a single client can be determined by sampling the statistics before and after that client sent a message.

A simplistic approach to producing safer statistics in such cases is to truncate and/or obfuscate the least significant bits of the statistics. It is often necessary to tailor such truncation to the types of measurements, e.g., number of requests is typically a very large number while the number of specific errors is usually small. Truncation could of course be done dynamically. More generally, the set of information provided to the operator about the confidential process could be viewed in light of differential privacy.

Another complementary approach is to provide statistics only at set intervals, or after a sufficient amount of new traffic has been received.

Another complementary technique to monitor the health of confidential services is the use of probes to ensure that the services function correctly. Probes can also measure the performance of the services.

The case of excessive service conditions due to Denial-of-Service attacks is discussed further under the Security Considerations section.

7.2. Debugging

Various error conditions and software issues may occur, as is usual with any service. There is a need to monitor problems that occur inside the service or at the client. This can be done, for instance, with the help of various statistics discussed earlier.

Some of the monitored conditions should include:

- * All major (or preferably even minor) error conditions should have an associated counter. This is necessary as no traditional logging can be reasonably provided that would otherwise have entries for, say, "client IP 203.0.113.0 sent a malformed request". While some errors can be expected at any time, a major increase in specific issues can indicate a problem. As a result, the counters need to be monitored and issues investigated as needed.
- * Client connection failures, which might indicate software version, trust root or other configuration problems.

Of course, for dedicated software testing purposes (such as debugging interoperability problems), even confidential services need to be run in a mode that exposes everything. Actual clients and users **MUST** be able to ensure that they are connected to a production service instance. This can be done by providing debugging status as part of the remote attestation, so that clients can verify it is off. Alternatively, testing versions of the service are simply not listed as trusted software versions.

7.3. Dependencies

The use of Confidential Computing introduces three additional dependencies to the system:

There is a need to be able to verify that the CPU executing the service is a legitimate CPU with the right hardware, and that the software being run for the service is acceptable. While this can be hard coded information in the service clients, in practice there is often a need to rely on other parties for scalability. As a result, there are two dependencies for legitimate CPU verification and for checking acceptable software versions. These are services that need to be run, and/or their use need to be agreed and possibly contracted for. The CPU manufacturer often plays a role in the CPU verification.

The third dependency is on the client. Depending on specific protocol arrangements, Confidential Computing services often can

serve unmodified clients, but for the full benefits and for validating attestations or software images, client changes are necessary. The necessary communications may happen as part of TLS negotiations or other general purpose protocols [I-D.mandyam-tokbind-attest], [I-D.ietf-rats-eat].

7.4. Additional services

Many services employ information that can be used to perform additional services beyond the basic task. For instance, knowledge about what the users requests or who the user is can be used for various optimizations or additional information that can be delivered to the user. Or the user can provide some additional information that is taken into account by the service.

One concern with these types of additional services is that the information used by them can be privacy sensitive. But Confidential Computing can assist in this as well, as long as the relevant information stays only within the TEE, it is better protected than by, e.g., providing that extra information to a regular service on the Internet.

Conversely, care needs to be taken whenever the service needs to relay some information outside the TEE. Some specific situations where this is needed with DNS are discussed in Section 7.1.

One example of additional services is that aggregate, privacy-sensitive data may be produced about trends in a confidentially run service, if it will not be possible to separate individual users from that data. For instance, it would be difficult sell information about individual users to help with targeted advertising, but the overall popularity of some websites could be measured.

7.5. Performance

Confidential Computing technology may impact performance. Nakatsuka et al. [PDoT] report on DNS resolution within a TEE where their solution could outperform the open source Unbound DNS server in certain scenarios, especially in situations where there are not a lot of DNS client connections. We concur their suggestion that at current stage of Confidential Computing technology, possible implementations may be more suited for local DNS resolution services rather than global scale implementation, where the performance hit would be much more significant. Nonetheless with Confidential Computing technology ever evolving we believe the low performance overhead solutions will be possible in foreseeable future.

Other things being equal there's likely some performance hit, as current Confidential Computing technology typically involves separating a server into two parts, the trusted and untrusted parts. In practice, all communications need to go through both, and the communication between the two parts consumes some cycles. There are also current limitations on amount of memory or threads supported by these technologies. However, newer virtualization-based confidential computing TEE approaches are likely going to improve these aspects.

Another performance hit comes from the overhead related to running the attestation process, and passing the necessary extra information in the communications protocols with the clients. In general, this works best when the cost of the setup is amortized over a long-lived session. Such sessions may exist between DoT/DoH-enabled clients and resolvers. Also, there are many possible arrangements and possible parties involved in attestation, see [I-D.ietf-rats-architecture].

8. Security Considerations

Security issues in this arrangement are discussed below.

8.1. Observations from outside the TEE

While a TEE is considered to be secure and not observable, there may be signs outside the TEE that can reveal information.

For instance, a server may receive a request from a client and immediately send out a question to a server in the Internet about a particular domain name. Observers - such as the owner of the server computer or the cloud farm - may be able to link incoming user queries to outgoing questions

Caching, randomly made other traffic, and timing obfuscation can deter such attacks, at least to an extent.

8.2. Trust Relationships

For scaling reasons, the arrangement typically depends on the ability to have trusted parties (a) for attesting the validity of a particular CPU being manufactured by a CPU manufacturer, and (b) for determining whether a particular software image hash is acceptable for the task it is advertising to do.

Such trusted parties need to be configured, which presents an additional operational burden. The information can of course be provided as part of a device manufacturer's or application's initial configuration, or be provided independently similar to how, for instance, certificate authorities are run.

It is important to recognize that mere use of technology is not sufficient to make the system secure. With communications, establishing a secure, encrypted channel is of no use if it is not with the intended party due to a certificate authority that proved to be untrustworthy. With confidential computing, the same applies: one has to have someone who can assert that a CPU is capable of performing the confidential computing task and that the indicated software is good for performing the task that the user expects it to perform. That being said, when such trusted parties can be found, the service performed by the server can become much more privacy friendly.

8.3. Denial-of-Service Attacks

To paraphrase an old philosophical question, "If an evil packet is sent behind the veil of encryption and no one is around to lift it, did an attack happen?" [Chautauquan]

Denial-of-Service attacks are a more serious form of the problems with operating services that the operator (intentionally) does not fully see. There needs to be means to deal with these attacks.

Attacks that can be identified by particularly high traffic flows from externally observable sources (e.g., source IP address) can of course still be dealt with in similar ways as we do in more open server designs.

But this is often not enough, and for this purpose some additional support is needed in the systems, for both detection of attacks and reacting to them.

One detection technique is to use the aggregate/truncated statistics to analyze anomalous behaviour. Another technique is to have the confidential part of the service produce extra information about events that cross a threshold. For instance, a particular error may occur exceptionally frequently, say among millions of requests, and this could warrant exposing either something about the request (e.g., the associated domain name) or something about the client (e.g., connection type, protocol details, or sender address).

The operator of the services needs to be able to react to possible attacks as well. One technique is to be able to provide instruction to the confidential part of the service to refuse service for specific requests (e.g., specific domain names) or for specific clients (e.g., coming from specific addresses). Alternatively, the service can also dynamically react to issues, e.g., by starting to reduce the amount of resources dedicated to some classes of requests that for some reason are starting to require exceptionally high

amount of resources. These techniques do not endanger user privacy, but may of course impact provided service.

8.4. Other vulnerabilities

Like all security mechanisms, this solution is not a panacea. It relies on the correct operation of a number of technologies and entities. For instance, CPU bugs or side channel vulnerabilities can cause information leaks to become possible. While confidential computing offers a layer of protection against attacks even from the owner of the computer hardware or the operating system, it is believed that this protection does not extend to sophisticated physical attacks, such being able to study chips with an electron microscope.

And as discussed above, it is also critical to check what software is being run, as otherwise any possible benefit would be negated by the possibly negligent or nefarious actions the unchecked software makes.

The mechanism does offer an additional layer of defense, however. It allows some of the trust that we place on our cloud platform owners, CPUs, and software applications to be verified and controlled with technical means. It may have some remaining vulnerabilities, but we obviously already depend on, for instance, the correct operation of our computing platforms. As such, Confidential Computing works to reduce some of the vulnerabilities in this area.

It should also be a desirable feature for users. A service that offers Confidential Computing-based protection of user data and can show that its software does not leak user-specific information is likely going to be more attractive to users than one that provides no such assurances. Of course, overall user choice depends on many factors beyond privacy, such as cost, ease of use, switching costs, and so on.

There is also a danger of attacks or pressure from intelligence agencies that could result in, e.g., the use of unpublicized vulnerabilities in an attempt to dwarf the protections in Confidential Computing. This could be used to perform pervasive monitoring, for instance [RFC7258]. Even so, it is always beneficial to push the costs and difficulty for attackers. Requiring parties who perform pervasive monitoring to employ complex technical attacks rather than being able to request logs from a service provider significantly increases the difficulty and risk associated with such monitoring.

9. Recommendations

Data held by servers SHOULD receive at least as much security attention as communications do.

The authors would like to draw attention to the problem of data leaks, particularly for data in use, and RECOMMEND the application of all available tools to prevent inappropriate access to users' information.

This is particularly crucial for DNS resolution services that have the potential to learn user's browsing histories. But the principles apply also to other services.

While using Confidential Computing without other modifications to the service in question is possible, real benefits can only be realized when the actual service is built for the purpose of avoiding data leaks or user data capture. Systems may need to be tuned or modified, for instance they MUST NOT produce logs that would negate purpose of running them inside a TEE to begin with. Mechanisms SHOULD be found to enable debugging and the detection of fault situations and attacks, again without exposing private information relating to individual users.

Some computing services can proceed on their own and require no interaction with the rest of the world. These are easier to secure. Even then, care SHOULD be taken to avoid request-response timing to provide information useful for side-channel attacks. If so, the owner of the server hardware can not determine much about what was going on.

However, other services may require interaction with other systems, such as is the case with a DNS resolver needing to find out a particular name that is not in a cache or whose cache entry has expired. This is because the resolution service is not a self-contained computation task but ultimately needs, at least in some cases, interaction with the rest of the world.

Consequently, the resolver needs to collaborate with other network nodes that are not even in the same administrative domain and cannot be guaranteed to subscribe to the same principles of protecting user's information. In this case, even if communications to other entities are encrypted, the potentially untrusted party at the other end of the communications may leak information.

In such communications, care SHOULD be taken to avoid exposing any information that would identify users, or allow fingerprinting the capabilities of those users' systems. Similarly, care SHOULD be

taken to avoid exposing any timing information that would allow the owner of the server hardware to determine what is going on, e.g., which users are asking for what names. Even so, vulnerabilities may appear if the attacker can force the system to behave in a particular way, by, e.g., forcing cache overflow, overloading it with traffic it knows about, etc.

The situation is slightly different when the interaction is with other systems that form a part of the same administrative domain. In particular, if those other systems employ similar confidential computing setup, and an encrypted channel is used, then some additional security can be provided compared to communicating with other entities in the Internet.

10. Acknowledgments

The authors would like to thank Juhani Kauppi, Jimmy Kjaellman, and Tero Kauppinen for their work on systems supporting some of the ideas discussed in this memo, and Dave Thaler, Daniel Migault, Karl Norrman, and Christian Schaefer for significant feedback on early version of this draft. The author would also like to thank Marcus Ihlar, Maria Luisa Mas, Miguel Angel Munos De La Torre Alonso, Jukka Ylitalo, Bengt Sahlin, Tomas Mecklin, Ben Smeets and many others for interesting discussions in this problem space.

11. References

11.1. Normative References

- [RFC1035] Mockapetris, P.V., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [AMD] Kaplan, D., Powell, J., and T. Woller, "AMD Memory Encryption", AMD White Paper , April 2016.

[Cambridge]

Isaak, J. and M. Hanna, "User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection", Computer 51.8 (2018): 56-59, <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8436400> , 2018.

[CC]

Rashid, F.Y., "What Is Confidential Computing?", IEEE Spectrum, <https://spectrum.ieee.org/computing/hardware/what-is-confidential-computing> , May 2020.

[CCC-Deepdive]

Confidential Computing Consortium, ., "A Technical Analysis of Confidential Computing", <https://confidentialcomputing.io/whitepaper-02-latest> , January 2021.

[Chautauquan]

"The Chautauquan", Volume 3, Issue 9, p. 543 , June 1883.

[Comparison]

Mofrad, S., Zhang, F., Lu, S., and W. Shi, "A comparison study of intel SGX and AMD memory encryption technology", HASP '18, Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, Pages 1-8, <https://doi.org/10.1145/3214292.3214301> , June 2018.

[Digging]

Hammouchi, H., Cherqi, O., Mezzour, G., Ghogho, M., and M. El Koutbi, "Digging Deeper into Data Breaches: An Exploratory Data Analysis of Hacking Breaches Over Time", Procedia Computer Science, Volume 151, pp. 1004-1009, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2019.04.141>, <https://www.sciencedirect.com/science/article/pii/S1877050919306064> , 2019.

[Efficient]

Suh, G.E., Clarke, D., Gasend, B., van Dijk, M., and S. Devadas, "Efficient memory integrity verification and encryption for secure processors", Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-36, San Diego, CA, USA, pp. 339-350, doi: 10.1109/MICRO.2003.1253207 , 2003.

[I-D.arkko-arch-infrastructure-centralisation]

Arkko, J., "Centralised Architectures in Internet Infrastructure", Work in Progress, Internet-Draft, draft-arkko-arch-infrastructure-centralisation-00, 4 November

2019, <<https://www.ietf.org/archive/id/draft-arkko-arch-infrastructure-centralisation-00.txt>>.

[I-D.arkko-farrell-arch-model-t-redux]

Arkko, J. and S. Farrell, "Internet Threat Model Evolution: Background and Principles", Work in Progress, Internet-Draft, draft-arkko-farrell-arch-model-t-redux-01, 22 February 2021, <<https://www.ietf.org/archive/id/draft-arkko-farrell-arch-model-t-redux-01.txt>>.

[I-D.iab-dedr-report]

Arkko, J. and T. Hardie, "Report from the IAB Workshop on Design Expectations vs. Deployment Reality in Protocol Development", Work in Progress, Internet-Draft, draft-iab-dedr-report-01, 2 November 2020, <<https://www.ietf.org/archive/id/draft-iab-dedr-report-01.txt>>.

[I-D.ietf-dprive-dnssoquic]

Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-ietf-dprive-dnssoquic-02, 22 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-dprive-dnssoquic-02.txt>>.

[I-D.ietf-rats-architecture]

Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-12, 23 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-rats-architecture-12.txt>>.

[I-D.ietf-rats-eat]

Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-10, 7 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-rats-eat-10.txt>>.

[I-D.ietf-tls-esni]

Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-11, 14 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-esni-11.txt>>.

[I-D.lazanski-smart-users-internet]

Lazanski, D., "An Internet for Users Again", Work in

Progress, Internet-Draft, draft-lazanski-smart-users-internet-00, 8 July 2019,
<<https://www.ietf.org/archive/id/draft-lazanski-smart-users-internet-00.txt>>.

[I-D.mandyam-tokbind-attest]

Mandyam, G., Lundblade, L., and J. Azen, "Attested TLS Token Binding", Work in Progress, Internet-Draft, draft-mandyam-tokbind-attest-07, 24 January 2019,
<<https://www.ietf.org/archive/id/draft-mandyam-tokbind-attest-07.txt>>.

[I-D.reddy-add-server-policy-selection]

Reddy, T., Wing, D., Richardson, M. C., and M. Boucadair, "DNS Server Selection: DNS Server Information with Assertion Token", Work in Progress, Internet-Draft, draft-reddy-add-server-policy-selection-08, 29 March 2021,
<<https://www.ietf.org/archive/id/draft-reddy-add-server-policy-selection-08.txt>>.

[I-D.thomson-tmi]

Thomson, M., "Principles for the Involvement of Intermediaries in Internet Protocols", Work in Progress, Internet-Draft, draft-thomson-tmi-01, 3 January 2021,
<<https://www.ietf.org/archive/id/draft-thomson-tmi-01.txt>>.

[I-D.voit-rats-attestation-results]

Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-voit-rats-attestation-results-01, 10 June 2021,
<<https://www.ietf.org/archive/id/draft-voit-rats-attestation-results-01.txt>>.

[Innovative]

Ittai, A., Gueron, S., Johnson, S., and V. Scarlata, "Innovative Technology for CPU Based Attestation and Sealing", HASP'2013 , 2013.

[Mem]

Henson, M. and S. Taylor, "Memory encryption: a survey of existing techniques", ACM Computing Surveys volume 46 issue 4 , 2014.

[MozTRR]

Mozilla, ., "Security/DOH-resolver-policy", <https://wiki.mozilla.org/Security/DOH-resolver-policy> , 2019.

- [PDoT] Nakatsuka, Y., Paverd, A., and G. Tsudik, "PDoT: Private DNS-over-TLS with TEE Support", Digit. Threat.: Res. Pract., Vol. 2, No. 1, Article 3, <https://dl.acm.org/doi/fullHtml/10.1145/3431171> , February 2021.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <<https://www.rfc-editor.org/info/rfc7626>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8324] Klensin, J., "DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?", RFC 8324, DOI 10.17487/RFC8324, February 2018, <<https://www.rfc-editor.org/info/rfc8324>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8558] Hardie, T., Ed., "Transport Protocol Path Signals", RFC 8558, DOI 10.17487/RFC8558, April 2019, <<https://www.rfc-editor.org/info/rfc8558>>.
- [RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <<https://www.rfc-editor.org/info/rfc8932>>.
- [SGX] Hoekstra, M.E., "Intel(R) SGX for Dummies (Intel(R) SGX Design Objectives)", Intel, <https://software.intel.com/content/www/us/en/develop/blogs/protecting-application-secrets-with-intel-sgx.html> , September 2013.

- [SmartTV] Malkin, N., Bernd, J., Johnson, M., and S. Egelman, "What Can't Data Be Used For? Privacy Expectations about Smart TVs in the U.S.", European Workshop on Usable Security (Euro USEC), https://www.ndss-symposium.org/wp-content/uploads/2018/06/eurousec2018_16_Malkin_paper.pdf , 2018.
- [Stallman] Stallman, R., "Can You Trust Your Computer?", GNU.org, <https://www.gnu.org/philosophy/can-you-trust.html> , n.d..
- [Toys] Chu, G., Apthorpe, N., and N. Feamster, "Security and Privacy Analyses of Internet of Things Childrens' Toys", IEEE Internet of Things Journal 6.1 (2019): 978-985, <https://arxiv.org/pdf/1805.02751.pdf> , 2019.
- [Unread] Obar, J. and A. Oeldorf, "The biggest lie on the internet{:} Ignoring the privacy policies and terms of service policies of social networking services", Information, Communication and Society (2018): 1-20 , 2018.
- [Vastaamo] Redcross Finland, ., "Read this if your personal data was leaked in the Vastaamo data system break-in", <https://www.redcross.fi/news/20201029/read-if-your-personal-data-was-leaked-vastaamo-data-system-break> , October 2020.

Authors' Addresses

Jari Arkko
Ericsson

Email: jari.arkko@ericsson.com

Jiri Novotny
Ericsson

Email: jiri.novotny@ericsson.com

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 7 November 2021

W. Hardaker
USC/ISI
V. Dukhovni
Bloomberg, L.P.
6 May 2021

Guidance for NSEC3 parameter settings
draft-hardaker-dnsop-nsec3-guidance-03

Abstract

NSEC3 is a DNSSEC mechanism providing proof of non-existence by promising there are no names that exist between two domainnames within a zone. Unlike its counterpart NSEC, NSEC3 avoids directly disclosing the bounding domainname pairs. This document provides guidance on setting NSEC3 parameters based on recent operational deployment experience.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 November 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Requirements notation	3
2. Recommendation for zone publishers	3
2.1. Algorithms	3
2.2. Flags	3
2.3. Iterations	3
2.4. Salt	4
3. Best-practice for zone publishers	5
4. Recommendation for validating resolvers	5
5. Security Considerations	6
6. Operational Considerations	6
7. IANA Considerations	6
8. References	6
8.1. Normative References	6
8.2. Informative References	7
Appendix A. Acknowledgments	7
Appendix B. Github Version of this document	7
Authors' Addresses	7

1. Introduction

As with NSEC [RFC4035], NSEC3 [RFC5155] provides proof of non-existence that consists of signed DNS records establishing the non-existence of a given name or associated Resource Record Type (RRTYPE) in a DNSSEC [RFC4035] signed zone. In the case of NSEC3, however, the names of valid nodes in the zone are obfuscated through (possibly multiple iterations of) hashing via SHA-1. (currently only SHA-1 is in use within the Internet).

NSEC3 also provides "opt-out support", allowing for blocks of unsigned delegations to be covered by a single NSEC3 record. Use of the opt-out feature allow large registries to only sign as many NSEC3 records as there are signed DS or other RRsets in the zone - with opt-out, unsigned delegations don't require additional NSEC3 records. This sacrifices the tamper-resistance proof of non-existence offered by NSEC3 in order to reduce memory and CPU overheads.

NSEC3 records have a number of tunable parameters that are specified via an NSEC3PARAM record at the zone apex. These parameters are the Hash Algorithm, processing Flags, the number of hash Iterations and the Salt. Each of these has security and operational considerations that impact both zone owners and validating resolvers. This document provides some best-practice recommendations for setting the NSEC3 parameters.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Recommendation for zone publishers

The following sections describe recommendations for setting parameters for NSEC3 and NSEC3PARAM.

2.1. Algorithms

The algorithm field is not discussed by this document.

2.2. Flags

The NSEC3PARAM flags field currently contains no flags, but individual NSEC3 records contain the "Opt-Out" flag [RFC5155], which specifies whether or not that NSEC3 record provides proof of non-existence or not. In general, NSEC3 with the Opt-Out flag enabled should only be used in large, highly dynamic zones with a small percentage of signed delegations. Operationally, this allows for fewer signature creations when new delegations are inserted into a zone. This is typically only necessary for extremely large registration points providing zone updates faster than real-time signing allows. Smaller zones, or large but relatively static zones, are encouraged to use a Flags value of 0 (zero) and take advantage of DNSSEC's proof-of-non-existence support.

2.3. Iterations

NSEC3 records are created by first hashing the input domain and then repeating that hashing algorithm a number of times based on the iterations parameter in the NSEC3PARAM and NSEC3 records. The first hash is typically sufficient to discourage zone enumeration performed by "zone walking" an NSEC or NSEC3 chain. Only determined parties with significant resources are likely to try and uncover hashed

values, regardless of the number of additional iterations performed. If an adversary really wants to expend significant CPU resources to mount an offline dictionary attack on a zone's NSEC3 chain, they'll likely be able to find most of the "guessable" names despite any level of additional hashing iterations.

Most names published in the DNS are rarely secret or unpredictable. They are published to be memorable, used and consumed by humans. They are often recorded in many other network logs such as email logs, certificate transparency logs, web page links, intrusion detection systems, malware scanners, email archives, etc. Many times a simple dictionary of commonly used domain names prefixes (www, ftp, mail, imap, login, database, etc) can be used to quickly reveal a large number of labels within a zone. Because of this, there are increasing performance costs yet diminishing returns associated with applying additional hash iterations beyond the first.

Although Section 10.3 of [RFC5155] specifies upper bounds for the number of hash iterations to use, there is no published guidance for zone owners about good values to select. Because hashing provides only moderate protection, as shown recently in academic studies of NSEC3 protected zones [GPUNSEC3][ZONEENUM], this document recommends that zone owners SHOULD use an iteration value of 0 (zero), indicating that only the initial hash value should be placed into a DNS zone's NSEC3 records.

TODO: discuss the authoritative overhead of needing to find the right range for new random strings coming in. Note white-lies online signing differences.

2.4. Salt

Salts add yet another layer of protection against offline, stored dictionary attacks by combining the value to be hashed (in our case, a DNS domainname) with a randomly generated value. This prevents adversaries from building up and remembering a dictionary of values that can translate a hash output back to the value that it derived from.

In the case of DNS, it should be noted the hashed names placed in NSEC3 records already include the fully-qualified domain name from each zone. Thus, no single pre-computed table works to speed up dictionary attacks against multiple target zones. An attacker is required to compute a complete dictionary per zone, which is expensive in both storage and CPU time.

To protect against a dictionary being built and used for a target zone, an additional salt field can be included and changed on a regular basis, forcing a would-be attacker to repeatedly compute a new dictionary (or just do trial and error without the benefits of precomputation).

Changing a zone's salt value requires the construction of a complete new NSEC3 chain. This is true both when resigning the entire zone at once, or incrementally signing it in the background where the new salt is only activated once every name in the chain has been completed.

Most users of NSEC3 publish static salt values that never change. This provides no added security benefit (because the complete fully qualified domain name is already unique). If no rotation is planned, operators are encouraged to forgo the salt entirely by using a zero-length salt value instead (represented as a "-" in the presentation format).

3. Best-practice for zone publishers

In short, for all zones, the recommended NSEC3 parameters are as shown below:

```
; SHA-1, no extra iterations, empty salt:
;
bcp.example. IN NSEC3PARAM 1 0 0 -
```

For small zones, the use of opt-out based NSEC3 records is NOT RECOMMENDED.

For very large and sparsely signed zones, where the majority of the records are insecure delegations, opt-out MAY be used.

4. Recommendation for validating resolvers

Because there has been a large growth of open (public) DNSSEC validating resolvers that are subject to compute resource constraints when handling requests from anonymous clients, this document recommends that validating resolvers should change their behavior with respect to large iteration values. Validating resolvers SHOULD return an insecure response when processing NSEC3 records with iterations larger than 100. Validating resolvers MAY return SERVFAIL when processing NSEC3 records with iterations larger than 500. Note that this significantly decreases the requirements originally specified in Section 10.3 of [RFC5155].

Note that a validating resolver MUST still validate the signature over the NSEC3 record to ensure the iteration count was not altered since record publication (see [RFC5155] section 10.3).

Validating resolvers returning an insecure or SERVFAIL answer in this situation SHOULD return an Extended DNS Error (EDE) {RFC8914} EDNS0 option of value [TBD].

5. Security Considerations

This entire document discusses security considerations with various parameters selections of NSEC3 and NSEC3PARAM fields.

6. Operational Considerations

This entire document discusses operational considerations with various parameters selections of NSEC3 and NSEC3PARAM fields.

7. IANA Considerations

This document requests a new allocation in the "Extended DNS Error Codes" of the "Domain Name System (DNS) Parameters" registration table with the following characteristics:

- * INFO-CODE: TBD
- * Purpose: Unsupported NSEC3 iterations value
- * Reference: this document

8. References

8.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.

8.2. Informative References

- [GPUNSEC3] Wander, M., Schwittmann, L., Boelmann, C., and T. Weis, "GPU-Based NSEC3 Hash Breaking", DOI 10.1109/NCA.2014.27, 2014, <<https://doi.org/10.1109/NCA.2014.27>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [ZONEENUM] Wang, Z., Xiao, L., and R. Wang, "An efficient DNSSEC zone enumeration algorithm", n.d..

Appendix A. Acknowledgments

dns-operations discussion participants

Appendix B. Github Version of this document

While this document is under development, it can be viewed, tracked, issued, pushed with PRs, ... here:

<https://github.com/hardaker/draft-hardaker-dnsop-nsec3-guidance>

Authors' Addresses

Wes Hardaker
USC/ISI

Email: ietf@hardakers.net

Viktor Dukhovni
Bloomberg, L.P.

Email: ietf-dane@dukhovni.org

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 27 June 2022

K. Fujiwara
JPRS
P. Vixie
none
24 December 2021

Fragmentation Avoidance in DNS
draft-ietf-dnsop-avoid-fragmentation-06

Abstract

EDNS0 enables a DNS server to send large responses using UDP and is widely deployed. Path MTU discovery remains widely undeployed due to security issues, and IP fragmentation has exposed weaknesses in application protocols. Currently, DNS is known to be the largest user of IP fragmentation. It is possible to avoid IP fragmentation in DNS by limiting response size where possible, and signaling the need to upgrade from UDP to TCP transport where necessary. This document proposes to avoid IP fragmentation in DNS.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Proposal to avoid IP fragmentation in DNS	3
3.1. Recommendations for UDP responders	4
3.2. Recommendations for UDP requestors	4
3.3. Default Maximum DNS/UDP payload size	4
4. Incremental deployment	6
5. Request to zone operators and DNS server operators	6
6. Considerations	6
6.1. Protocol compliance	6
7. IANA Considerations	7
8. Security Considerations	7
9. Acknowledgments	7
10. References	7
10.1. Normative References	7
10.2. Informative References	8
Appendix A. Weaknesses of IP fragmentation	9
Appendix B. Details of maximum DNS/UDP payload size discussions	10
Appendix C. How to retrieve path MTU value to a destination from applications	11
Appendix D. How to retrieve minimal MTU value to a destination	11
Appendix E. Minimal-responses	11
Authors' Addresses	12

1. Introduction

DNS has EDNS0 [RFC6891] mechanism. It enables a DNS server to send large responses using UDP. EDNS0 is now widely deployed, and DNS (over UDP) is said to be the biggest user of IP fragmentation.

Fragmented DNS UDP responses have systemic weaknesses, which expose the requestor to DNS cache poisoning from off-path attackers. (See Appendix A for references and details.)

[RFC8900] summarized that IP fragmentation introduces fragility to Internet communication. The transport of DNS messages over UDP should take account of the observations stated in that document.

TCP avoids fragmentation using its Maximum Segment Size (MSS) parameter, but each transmitted segment is header-size aware such that the size of the IP and TCP headers is known, as well as the far end's MSS parameter and the interface or path MTU, so that the segment size can be chosen so as to keep the each IP datagram below a target size. This takes advantage of the elasticity of TCP's packetizing process as to how much queued data will fit into the next segment. In contrast, DNS over UDP has little datagram size elasticity and lacks insight into IP header and option size, and so must make more conservative estimates about available UDP payload space.

This document proposes to set IP_DONTFRAG / IPV6_DONTFRAG in DNS/UDP messages in order to avoid IP fragmentation, and describes how to avoid packet losses due to IP_DONTFRAG / IPV6_DONTFRAG.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"Requestor" refers to the side that sends a request. "Responder" refers to an authoritative, recursive resolver or other DNS component that responds to questions. (Quoted from EDNS0 [RFC6891])

"Path MTU" is the minimum link MTU of all the links in a path between a source node and a destination node. (Quoted from [RFC8201])

"Path MTU discovery" is defined by [RFC1191], [RFC8201] and [RFC8899].

IP_DONTFRAG option is not defined by any RFCs. It is similar to IPV6_DONTFRAG option defined in [RFC3542]. IP_DONTFRAG option is used on BSD systems to set the Don't Fragment bit [RFC0791] when sending IPv4 packets. On Linux systems this is done via IP_MTU_DISCOVER and IP_PMTUDISC_DO.

Many of the specialized terms used in this document are defined in DNS Terminology [RFC8499].

3. Proposal to avoid IP fragmentation in DNS

The methods to avoid IP fragmentation in DNS are described below:

3.1. Recommendations for UDP responders

- * UDP responders SHOULD send DNS responses with IP_DONTFRAG / IPV6_DONTFRAG [RFC3542] options.
- * If the UDP responder detects immediate error that the UDP packet cannot be sent beyond the path MTU size (EMSGSIZE), the UDP responder MAY recreate response packets fit in path MTU size, or TC bit set.
- * UDP responders MAY probe to discover the real MTU value per destination.
- * UDP responders SHOULD compose UDP responses that result in IP packets that do not exceed the path MTU to the requestor. If the path MTU discovery failed or is impossible, UDP responders SHOULD compose UDP responses that result in IP packets that do not exceed the default maximum DNS/UDP payload size described in Section 3.3.

The cause and effect of the TC bit is unchanged from EDNS0 [RFC6891].

3.2. Recommendations for UDP requestors

- * UDP requestors SHOULD send DNS requests with IP_DONTFRAG / IPV6_DONTFRAG [RFC3542] options.
- * UDP requestors MAY probe to discover the real MTU value per destination. Then, calculate their maximum DNS/UDP payload size as the reported path MTU minus IPv4/IPv6 header size (20 or 40) minus UDP header size (8). If the path MTU discovery failed or is impossible, use the default maximum DNS/UDP payload size described in Section 3.3.
- * UDP requestors SHOULD use the requestor's payload size as the calculated or the default maximum DNS/UDP payload size.
- * UDP requestors MAY drop fragmented DNS/UDP responses without IP reassembly to avoid cache poisoning attacks.
- * DNS responses may be dropped by IP fragmentation. Upon a timeout, UDP requestors may retry using TCP or UDP, per local policy.

3.3. Default Maximum DNS/UDP payload size

Fragmentation avoidance is achieved with the IP(V6)_DONTFRAG option. The purpose of packet size limitation is to decrease packet loss due to the effects of the IP(V6)_DONTFRAG option.

Default maximum DNS/UDP payload size depends on the connectivity of each node, it cannot be determined unconditionally. However, there are good proposed values.

Operators MAY select a good number from Table 1. Details of proposed values are described in Appendix B.

Source	IPv4	IPv6
Minimal: RFC 4035 MUST	1220	1220
Software developers / DNSFlagDay2020 propose	1232	1232 (1280-40-8)
Authors' recommendation	1400	1400 (1500 -40 -8 - some headers)
Maximum: Ethernet MTU 1500 [Huston2021]	1472 (1500-20-8)	1452 (1500-40-8)
Measured	MTU -20-8	MTU -40-8

Table 1: Default maximum DNS/UDP payload size

However, operators of DNS servers SHOULD measure their path MTU to the Internet at setting up DNS servers (and when network configuration changes).

How to measure path MTU is described in Appendix D.

Operators of authoritative servers (that offer global DNS zones) and full-service resolvers (that access authoritative servers of the global DNS) SHOULD measure their path MTU to well-known locations on the Internet, such as [a-m].root-servers.net or [a-m].gtld-servers.net.

Operators of full-service resolvers would be well advised to measure their path MTU to several authority name servers and to a random sample of their expected stub resolver client networks, to find the upper boundary on IP/UDP packet size in the average case. Or, operators of ISPs know their customers' connectivity and customers' MTU to ISPs' servers. This limit should not be exceeded by most messages received or transmitted by a full resolver, or else fallback to TCP will occur too often.

DNS clients (stub resolvers) need to specify an appropriate requestor's payload size when supporting EDNS0. In case of CPEs, embedded devices, and user devices, network operators can not control them, developers may choose small values such as 1220 and 1232.

Other DNS servers are out-of-scope of this document. (For example, Forwarding only resolvers, or private DNS).

4. Incremental deployment

The proposed method supports incremental deployment.

When a full-service resolver implements the proposed method, its stub resolvers (clients) and the authority server network will no longer observe IP fragmentation or reassembly from that server, and will fall back to TCP when necessary.

When an authoritative server implements the proposed method, its full service resolvers (clients) will no longer observe IP fragmentation or reassembly from that server, and will fall back to TCP when necessary.

5. Request to zone operators and DNS server operators

Large DNS responses are the result of zone configuration. Zone operators SHOULD seek configurations resulting in small responses. For example,

- * Use smaller number of name servers (13 may be too large)
- * Use smaller number of A/AAAA RRs for a domain name
- * Use 'minimal-responses' configuration: Some implementations have 'minimal responses' configuration that causes DNS servers to make response packets smaller, containing only mandatory and required data (Appendix E).
- * Use smaller signature / public key size algorithm for DNSSEC. Notably, the signature size of ECDSA or EdDSA is smaller than RSA.

6. Considerations

6.1. Protocol compliance

In prior research ([Fujiwara2018] and dns-operations mailing list discussions), there are some authoritative servers that ignore EDNS0 requestor's UDP payload size, and return large UDP responses.

It is also well known that there are some authoritative servers that do not support TCP transport.

Such non-compliant behavior cannot become implementation or configuration constraints for the rest of the DNS. If failure is the result, then that failure must be localized to the non-compliant servers.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

9. Acknowledgments

The author would like to specifically thank Paul Wouters, Mukund Sivaraman, Tony Finch, Hugo Salgado, Peter van Dijk, Brian Dickson, Puneet Sood and Jim Reid for extensive review and comments.

10. References

10.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <<https://www.rfc-editor.org/info/rfc3542>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.

- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <<https://www.rfc-editor.org/info/rfc8899>>.
- [RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.

10.2. Informative References

- [Brandt2018] Brandt, M., Dai, T., Klein, A., Shulman, H., and M. Waidner, "Domain Validation++ For MitM-Resilient PKI", Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security , 2018.
- [DNSFlagDay2020] "DNS flag day 2020", n.d., <<https://dnsflagday.net/2020/>>.
- [Fujiwara2018] Fujiwara, K., "Measures against cache poisoning attacks using IP fragmentation in DNS", OARC 30 Workshop , 2019.
- [Herzberg2013] Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", IEEE Conference on Communications and Network Security , 2013.

- [Hlavacek2013] Hlavacek, T., "IP fragmentation attack on DNS", RIPE 67 Meeting , 2013, <<https://ripe67.ripe.net/presentations/240-ipfragattack.pdf>>.
- [Huston2021] Huston, G. and J. Damas, "Measuring DNS Flag Day 2020", OARC 34 Workshop , February 2021.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<https://www.rfc-editor.org/info/rfc7739>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

Appendix A. Weaknesses of IP fragmentation

"Fragmentation Considered Poisonous" [Herzberg2013] proposed effective off-path DNS cache poisoning attack vectors using IP fragmentation. "IP fragmentation attack on DNS" [Hlavacek2013] and "Domain Validation++ For MitM-Resilient PKI" [Brandt2018] proposed that off-path attackers can intervene in path MTU discovery [RFC1191] to perform intentionally fragmented responses from authoritative servers. [RFC7739] stated the security implications of predictable fragment identification values.

DNSSEC is a countermeasure against cache poisoning attacks that use IP fragmentation. However, DNS delegation responses are not signed with DNSSEC, and DNSSEC does not have a mechanism to get the correct response if an incorrect delegation is injected. This is a denial-of-service vulnerability that can yield failed name resolutions. If cache poisoning attacks can be avoided, DNSSEC validation failures will be avoided.

In Section 3.2 (Message Side Guidelines) of UDP Usage Guidelines [RFC8085] we are told that an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination.

A DNS message receiver cannot trust fragmented UDP datagrams primarily due to the small amount of entropy provided by UDP port numbers and DNS message identifiers, each of which being only 16 bits in size, and both likely being in the first fragment of a packet, if fragmentation occurs. By comparison, TCP protocol stack controls packet size and avoid IP fragmentation under ICMP NEEDFRAG attacks. In TCP, fragmentation should be avoided for performance reasons, whereas for UDP, fragmentation should be avoided for resiliency and authenticity reasons.

Appendix B. Details of maximum DNS/UDP payload size discussions

There are many discussions for default path MTU size and maximum DNS/UDP payload size.

- * The minimum MTU for an IPv6 interface is 1280 octets (see Section 5 of [RFC8200]). Then, we can use it as default path MTU value for IPv6. The corresponding minimum MTU for an IPv4 interface is 68 (60 + 8) [RFC0791].
- * Most of the Internet and especially the inner core has an MTU of at least 1500 octets. Maximum DNS/UDP payload size for IPv6 on MTU 1500 ethernet is 1452 (1500 minus 40 (IPv6 header size) minus 8 (UDP header size)). To allow for possible IP options and distant tunnel overhead, authors' recommendation of default maximum DNS/UDP payload size is 1400.
- * [RFC4035] defines that "A security-aware name server MUST support the EDNS0 message size extension, MUST support a message size of at least 1220 octets". Then, the smallest number of the maximum DNS/UDP payload size is 1220.
- * In order to avoid IP fragmentation, [DNSFlagDay2020] proposed that the UDP requestors set the requestor's payload size to 1232, and the UDP responders compose UDP responses fit in 1232 octets. The

size 1232 is based on an MTU of 1280, which is required by the IPv6 specification [RFC8200], minus 48 octets for the IPv6 and UDP headers.

- * [Huston2021] analyzed the result of [DNSFlagDay2020], reported that their measurements suggest that in the interior of the Internet between recursive resolvers and authoritative servers the prevailing MTU is at 1,500 and there is no measurable signal of use of smaller MTUs in this part of the Internet, and proposed that their measurements suggest setting the EDNS0 Buffer size to IPv4 1472 octets and IPv6 1452 octets.

Appendix C. How to retrieve path MTU value to a destination from applications

Socket options: "IP_MTU (since Linux 2.2) Retrieve the current known path MTU of the current socket. Valid only when the socket has been connected. Returns an integer. Only valid as a getsockopt(2)." (Quoted from Debian GNU Linux manual: ip(7))

"IPV6_MTU getsockopt(): Retrieve the current known path MTU of the current socket. Only valid when the socket has been connected. Returns an integer." (Quoted from Debian GNU Linux manual: ipv6(7))

Section 3.4 of [RFC1122] specifies FIND_MAXSIZES() as one of "INTERNET/TRANSPORT LAYER INTERFACES".

Appendix D. How to retrieve minimal MTU value to a destination

The Linux tool "tracert" can be used to measure the path MTU to a destination.

Or, "ping/ping6" command with "-D" Don't Fragment bit set / Disable IPv6 fragmentation options.

Appendix E. Minimal-responses

Some implementations have 'minimal responses' configuration that causes a DNS server to make response packets smaller, containing only mandatory and required data.

Under the minimal-responses configuration, DNS servers compose response messages using only RRSets corresponding to queries. In case of delegation, DNS servers compose response packets with delegation NS RRSets in authority section and in-domain (in-zone and below-zone) glue in the additional data section. In case of non-existent domain name or non-existent type, the start of authority (SOA RR) will be placed in the Authority Section.

In addition, if the zone is DNSSEC signed and a query has the DNSSEC OK bit, signatures are added in answer section, or the corresponding DS RRSets and signatures are added in authority section. Details are defined in [RFC4035] and [RFC5155].

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

Paul Vixie
none
11400 La Honda Road
Woodside, CA, 94062
United States of America

Phone: +1 650 393 3994
Email: paul@redbarn.org

DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

P. van Dijk
PowerDNS
L. Peltan
CZ.NIC
O. Sury
Internet Systems Consortium
W. Toorop
NLnet Labs
K. Monshouwer

P. Thomassen
deSEC, SSE - Secure Systems Engineering
7 March 2022

DNS Catalog Zones
draft-ietf-dnsop-dns-catalog-zones-05

Abstract

This document describes a method for automatic DNS zone provisioning among DNS primary and secondary nameservers by storing and transferring the catalog of zones to be provisioned as one or more regular DNS zones.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Description	4
4. Catalog Zone Structure	4
4.1. SOA and NS Records	4
4.2. Member Zones	4
4.3. Global Properties	5
4.3.1. Schema Version (version property)	6
4.4. Member Zone Properties	6
4.4.1. Change of Ownership (coo property)	6
4.4.2. Groups (group property)	7
4.5. Custom Properties (*.ext properties)	8
5. Nameserver Behavior	9
5.1. General Requirements	9
5.2. Member zone name clash	9
5.3. Member zone removal	10
5.4. Member node name change	10
5.5. Migrating member zones between catalogs	10
5.6. Zone associated state reset	11
6. Implementation Notes	11
7. Security Considerations	11
8. Acknowledgements	12
9. Normative References	12
10. Informative References	13
Appendix A. Implementation Status	14
Appendix B. Change History (to be removed before final publication)	14
Authors' Addresses	17

1. Introduction

The content of a DNS zone is synchronized amongst its primary and secondary nameservers using AXFR and IXFR. However, the list of zones served by the primary (called a catalog in [RFC1035]) is not automatically synchronized with the secondaries. To add or remove a zone, the administrator of a DNS nameserver farm not only has to add or remove the zone from the primary, they must also add/remove the

zone from all secondaries, either manually or via an external application. This can be both inconvenient and error-prone; it is also dependent on the nameserver implementation.

This document describes a method in which the catalog is represented as a regular DNS zone (called a "catalog zone" here), and transferred using DNS zone transfers. As zones are added to or removed from the catalog zone, these changes are distributed to the secondary nameservers in the normal way. The secondary nameservers then add/remove/modify the zones they serve in accordance with the changes to the catalog zone. Other use-cases of nameserver remote configuration by catalog zones are possible, where the catalog consumer might not be a secondary.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Catalog zone A DNS zone containing a DNS catalog, that is, a list of DNS zones and associated properties.

Member zone A DNS zone whose configuration is published inside a catalog zone.

Member node The DNS name in the Catalog zone representing a Member zone.

\$CATZ Used in examples as a placeholder to represent the domain name of the catalog zone itself. \$OLDCATZ and \$NEWCATZ are used to discuss migration a member zone from one catalog zone \$OLDCATZ to another catalog zone \$NEWCATZ.

Catalog producer An entity that generates and is responsible for the contents of the catalog zone.

Catalog consumer An entity that extracts information from the catalog zone (such as a DNS server that configures itself according to the catalog zone's contents).

3. Description

A catalog zone is a DNS zone whose contents are specially crafted. Its records primarily constitute a list of PTR records referencing other DNS zones (so-called "member zones"). The catalog zone may contain other records indicating additional metadata (so-called "properties") associated with these member zones.

Catalog consumers SHOULD ignore any RR in the catalog zone which is meaningless or useless to the implementation.

Authoritative servers may be preconfigured with multiple catalog zones, each associated with a different set of configurations.

Although the contents of a catalog zone are interpreted and acted upon by nameservers, a catalog zone is a regular DNS zone and so must adhere to the standards for such zones.

A catalog zone is primarily intended for the management of a farm of authoritative nameservers. The content of catalog zones may not be accessible from any recursive nameserver.

4. Catalog Zone Structure

4.1. SOA and NS Records

As with any other DNS zone, a catalog zone MUST have a syntactically correct SOA record and at least one NS record at its apex.

The SOA record's SERIAL, REFRESH, RETRY and EXPIRE fields [RFC1035] are used during zone transfer. A catalog zone's SOA SERIAL field MUST increase when an update is made to the catalog zone's contents as per serial number arithmetic defined in [RFC1982]. Otherwise, catalog consumers might not notice updates to the catalog zone's contents.

There is no requirement to be able to query the catalog zone via recursive nameservers. Catalog consumers MUST ignore and MUST NOT assume or require NS records at the apex. However, at least one is still required so that catalog zones are syntactically correct DNS zones. A single NS RR with a NSDNAME field containing the absolute name "invalid." is RECOMMENDED [RFC2606].

4.2. Member Zones

The list of member zones is specified as a collection of member nodes, represented by domain names under the owner name "zones" where "zones" is a direct child domain of the catalog zone.

The names of member zones are represented on the RDATA side (instead of as a part of owner names) of a PTR record, so that all valid domain names may be represented regardless of their length [RFC1035]. This PTR record MUST be the only record in the PTR RRset with the same name. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

For example, if a catalog zone lists three zones "example.com.", "example.net." and "example.org.", the member node RRs would appear as follows:

```
<unique-1>.zones.$CATZ 0 IN PTR example.com.  
<unique-2>.zones.$CATZ 0 IN PTR example.net.  
<unique-3>.zones.$CATZ 0 IN PTR example.org.
```

where <unique-N> is a label that tags each record in the collection. <unique-N> has an unique value in the collection.

Member node labels carry no informational meaning beyond labeling member zones. A changed label may indicate that the state for a zone needs to be reset (see Section 5.6).

Having the zones uniquely tagged with the <unique-N> label ensures that additional RRs can be added below the member node (see Section 4.4).

The CLASS field of every RR in a catalog zone MUST be IN (1).

The TTL field's value is not defined by this memo. Catalog zones are for authoritative nameserver management only and are not intended for general querying via recursive resolvers.

4.3. Global Properties

Apart from catalog zone metadata stored at the apex (NS, SOA and the like), catalog zone information is stored in the form of "properties". Catalog consumers SHOULD ignore properties they do not understand.

This specification defines a number of so-called properties, as well as a mechanism to allow implementers to store additional information in the catalog zone with Custom properties, see Section 4.5. The meaning of such custom properties is determined by the implementation in question.

Some properties are defined at the global level; others are scoped to apply only to a specific member zone. This document defines a single mandatory global property in Section 4.3.1. Member-specific properties are described in Section 4.4.

More properties may be defined in future documents.

4.3.1. Schema Version (version property)

The catalog zone schema version is specified by an integer value embedded in a TXT RR named version.\$CATZ. All catalog zones MUST have a TXT RRset named version.\$CATZ with exactly one RR. Catalog consumers MUST NOT apply catalog zone processing to zones without the expected value in the version.\$CATZ TXT RR, but they may be transferred as ordinary zones. For this memo, the value of the version.CATZ TXT RR MUST be set to "2", i.e.:

```
version.$CATZ 0 IN TXT "2"
```

NB: Version 1 was used in a draft version of this memo and reflected the implementation first found in BIND 9.11.

4.4. Member Zone Properties

Each member zone MAY have one or more additional properties, described in this chapter. These properties are completely optional and catalog consumers SHOULD ignore those it does not understand. Member zone properties are represented by RRsets below the corresponding member node.

4.4.1. Change of Ownership (coo property)

The coo property facilitates controlled migration of a member zone from one catalog to another.

A Change Of Ownership is signaled by the coo property in the catalog zone currently "owning" the zone. The name of the new catalog is in the value of a PTR record in the old catalog. For example if member "example.com." will migrate from catalog zone \$OLDCATZ to catalog zone \$NEWCATZ, this appears in the \$OLDCATZ catalog zone as follows:

```
<unique-N>.zones.$OLDCATZ 0 IN PTR example.com.  
coo.<unique-N>.zones.$OLDCATZ 0 IN PTR $NEWCATZ
```

The PTR RRset MUST consist of a single PTR record. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

When a consumer of catalog zone \$OLDCATZ receives an update which adds or changes a coo property for a member zone in \$OLDCATZ signalling a new owner \$NEWCATZ, it does not migrate the member zone immediately.

This is because the catalog consumer may not have the <unique-N> identifier associated with the member zone in \$NEWCATZ and because name servers do not index Resource Records by RDATA, it may not know whether or not the member zone is configured in \$NEWCATZ at all. It may have to wait for an update of \$NEWCATZ adding or changing that member zone. When a consumer of catalog zone \$NEWCATZ receives an update of \$NEWCATZ which adds or changes a member zone, and that consumer had the member zone associated with \$OLDCATZ, and there is a coo property of the member zone in \$OLDCATZ pointing to \$NEWCATZ, only then it will reconfigure the member zone with the for \$NEWCATZ preconfigured settings.

Unless the member node label (i.e. <unique-N>) for the member is the same in \$NEWCATZ, all associated state for a just migrated zone **MUST** be reset (see Section 5.6). Note that the owner of \$OLDCATZ allows for the zone associated state to be taken over by the owner of \$NEWCATZ by default. To prevent the takeover, the owner of \$OLDCATZ has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneous with adding the coo property. (see also Section 7)

The old owner may remove the member zone containing the coo property from \$OLDCATZ once it has been established that all its consumers have processed the Change of Ownership.

4.4.2. Groups (group property)

With a group property, consumer(s) can be signalled to treat some member zones within the catalog zone differently.

The consumer **MAY** apply different configuration options when processing member zones, based on the value of the group property. The exact handling of configuration referred to by the group property value is left to the consumer's implementation and configuration. The property is defined by a TXT record in the sub-node labelled group.

The producer **MAY** assign a group property to all, some, or none of the member zones within a catalog zone. The producer **MUST NOT** assign more than one group property to one member zone.

The consumer **MUST** ignore either all or none of the group properties in a catalog zone.

The value of the TXT record MUST be at most 255 octets long and MUST NOT contain whitespace characters. The consumer MUST interpret the value case-sensitively.

4.4.2.1. Example

```
<unique-1>.zones.$CATZ      0 IN PTR      example.com.  
group.<unique-1>.zones.$CATZ 0 IN TXT      sign-with-nsec3  
<unique-2>.zones.$CATZ      0 IN PTR      example.net.  
group.<unique-2>.zones.$CATZ 0 IN TXT      nodnssec
```

In this case, the consumer might be implemented and configured in the way that the member zones with "nodnssec" group assigned will not be signed with DNSSEC, and the zones with "sign-with-nsec3" group assigned will be signed with DNSSEC with NSEC3 chain.

By generating the catalog zone (snippet) above, the producer signals how the consumer shall treat DNSSEC for the zones example.net. and example.com., respectively.

4.5. Custom Properties (*.ext properties)

Implementations and operators of catalog zones may choose to provide their own properties. Custom properties can occur both globally, or for a specific member zone. To prevent a name clash with future properties, such properties should be represented below the label ext.

ext is not a placeholder, so a custom property would have domains names as follows:

```
<your-property>.ext.$CATZ      # for a global custom property  
<your-property>.ext.<unique-N>.zones.$CATZ # for a member zone custom property
```

<your-property> may consist of one or more labels.

Implementations MAY use such properties on the member zone level to store additional information about member zones, for example to flag them for specific treatment (such as ...).

Further, implementations MAY use custom properties on the global level to store additional information about the catalog zone itself. While there may be many use cases for this, a plausible one is to store default values for custom properties on the global level, then overriding them using a property of the same name on the member level (= under the ext label of the member node) if so desired. A property description should clearly say what semantics apply, and whether a property is global, member, or both.

The meaning of the custom properties described in this section is determined by the implementation alone, without expectation of interoperability. A catalog consumer SHOULD ignore custom properties it does not understand.

5. Nameserver Behavior

5.1. General Requirements

As it is a regular DNS zone, a catalog zone can be transferred using DNS zone transfers among nameservers.

Although they are regular DNS zones, catalog zones contain only information for the management of a set of authoritative nameservers. For this reason, operators may want to limit the systems able to query these zones. It may be inconvenient to serve some contents of catalog zones via DNS queries anyway due to the nature of their representation. A separate method of querying entries inside the catalog zone may be made available by nameserver implementations (see Section 6).

Catalog updates should be automatic, i.e., when a nameserver that supports catalog zones completes a zone transfer for a catalog zone, it SHOULD apply changes to the catalog within the running nameserver automatically without any manual intervention.

As with regular zones, primary and secondary nameservers for a catalog zone may be operated by different administrators. The secondary nameservers may be configured as catalog consumer to synchronize catalog zones from the primary, but the primary's administrators may not have any administrative access to the secondaries.

Nameservers MAY allow loading and transfer of broken zones with incorrect catalog zone syntax (as they are treated as regular zones), but catalog consumers MUST NOT process such broken zones as catalog zones. For the purpose of catalog processing, the broken catalogs MUST be ignored.

5.2. Member zone name clash

If there is a clash between an existing zone's name (either from an existing member zone or otherwise configured zone) and an incoming member zone's name (via transfer or update), the new instance of the zone MUST be ignored and an error SHOULD be logged.

A clash between an existing member zone's name and an incoming member zone's name (via transfer or update), may be an attempt to migrate a zone to a different catalog, but should not be treated as one except as described in {#cooproperty}.

5.3. Member zone removal

When a member zone is removed from a specific catalog zone, an authoritative server **MUST NOT** remove the zone and associated state data if the zone was not configured from that specific catalog zone. Only when the zone was configured from a specific catalog zone, and the zone is removed as a member from that specific catalog zone, the zone and associated state (such as zone data and DNSSEC keys) **MUST** be removed.

5.4. Member node name change

When via a single update or transfer, the member node's label value (<unique-N>) changes, catalog consumers **MUST** process this as a member zone removal including all the zone's associated state (as described in Section 5.3), immediately followed by processing the member as a newly to be configured zone in the same catalog.

5.5. Migrating member zones between catalogs

If all consumers of the catalog zones involved support the cooproperty, it is **RECOMMENDED** to perform migration of a member zone by following the procedure described in Section 4.4.1. Otherwise a migration of member zone from a catalog zone \$OLDCATZ to a catalog zone \$NEWCATZ has to be done by: first removing the member zone from \$OLDCATZ; second adding the member zone to \$NEWCATZ.

If in the process of a migration some consumers of the involved catalog zones did not catch the removal of the member zone from \$OLDCATZ yet (because of a lost packet or down time or otherwise), but did already see the update of \$NEWCATZ, they may consider the update adding the member zone in \$NEWCATZ to be a name clash (see Section 5.2) and as a consequence the member is not migrated to \$NEWCATZ. This possibility needs to be anticipated with a member zone migration. Recovery from such a situation is out of the scope of this document. It may for example entail a manually forced retransfer of \$NEWCATZ to consumers after they have been detected to have received and processed the removal of the member zone from \$OLDCATZ.

5.6. Zone associated state reset

It may be desirable to reset state (such as zone data and DNSSEC keys) associated with a member zone.

A zone state reset may be performed by a change of the member node's name (see Section 5.4).

6. Implementation Notes

Catalog zones on secondary nameservers would have to be setup manually, perhaps as static configuration, similar to how ordinary DNS zones are configured. The secondary additionally needs to be configured as a catalog consumer for the catalog zone to enable processing of the member zones in the catalog, such as automatic synchronization of the member zones for secondary service.

An administrator may want to look at data inside a catalog zone. Typical queries might include dumping the list of member zones, dumping a member zone's effective configuration, querying a specific property value of a member zone, etc. Because of the structure of catalog zones, it may not be possible to perform these queries intuitively, or in some cases, at all, using DNS QUERY. For example, it is not possible to enumerate the contents of a multi-valued property (such as the list of member zones) with a single QUERY. Implementations are therefore advised to provide a tool that uses either the output of AXFR or an out-of-band method to perform queries on catalog zones.

7. Security Considerations

As catalog zones are transmitted using DNS zone transfers, it is RECOMMENDED that catalog zone transfer are protected from unexpected modifications by way of authentication, for example by using TSIG [RFC8945], or Strict or Mutual TLS authentication with DNS Zone transfer over TLS [RFC9103].

Use of DNS UPDATE [RFC2136] to modify the content of catalog zones SHOULD similarly be authenticated.

Zone transfers of member zones SHOULD similarly be authenticated. TSIG shared secrets used for member zones SHOULD NOT be mentioned in the catalog zone data. However, key identifiers may be shared within catalog zones.

Catalog zones reveal the zones served by the consumers of the catalog zone. It is RECOMMENDED to limit the systems able to query these zones. It is RECOMMENDED to transfer catalog zones confidentially [RFC9103].

Administrative control over what zones are served from the configured name servers shifts completely from the server operator (consumer) to the "owner" (producer) of the catalog zone content.

With migration of member zones between catalogs using the `coo` property, it is possible for the owner of the target catalog (i.e. `$NEWCATZ`) to take over all associated state with the zone from the original owner (i.e. `$OLDCATZ`) by maintaining the same member node label (i.e. `<unique-N>`). To prevent the takeover of the zone associated state, the original owner has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneously with adding the `coo` property.

8. Acknowledgements

Our deepest thanks and appreciation go to Stephen Morris, Ray Bellis and Witold Krecicki who initiated this draft and did the bulk of the work.

Catalog zones originated as the chosen method among various proposals that were evaluated at ISC for easy zone management. The chosen method of storing the catalog as a regular DNS zone was proposed by Stephen Morris.

The initial authors discovered that Paul Vixie's earlier [Metazones] proposal implemented a similar approach and reviewed it. Catalog zones borrows some syntax ideas from Metazones, as both share this scheme of representing the catalog as a regular DNS zone.

Thanks to Leo Vandewoestijne. Leo's presentation in the DNS devroom at the FOSDEM'20 [FOSDEM20] was one of the motivations to take up and continue the effort of standardizing catalog zones.

Thanks to Brian Conry, Klaus Darilion, Brian Dickson, Tony Finch, Evan Hunt, Shane Kerr, Patrik Lundin, Victoria Risk, Petr Spacek and Carsten Strotmann for reviewing draft proposals and offering comments and suggestions.

9. Normative References

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

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, <<https://www.rfc-editor.org/info/rfc2606>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <<https://www.rfc-editor.org/info/rfc8945>>.
- [RFC9103] Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer over TLS", RFC 9103, DOI 10.17487/RFC9103, August 2021, <<https://www.rfc-editor.org/info/rfc9103>>.

10. Informative References

- [FOSDEM20] Vandewoestijne, L., "Extending Catalog zones - another approach in automating maintenance", 2020, <https://archive.fosdem.org/2020/schedule/event/dns_catz/>.
- [Metazones] Vixie, P., "Federated Domain Name Service Using DNS Metazones", 2005, <<http://family.redbarn.org/~vixie/mz.pdf>>.

Appendix A. Implementation Status

Note to the RFC Editor: please remove this entire section before publication.

In the following implementation status descriptions, "DNS Catalog Zones" refers to DNS Catalog Zones as described in this document.

- * Knot DNS 3.1 (released August 2, 2021) supports full producing and consuming of catalog zones, including the group property.
- * PowerDNS has a proof of concept external program called PowerCATZ (<https://github.com/PowerDNS/powercatz/>), that can process DNS Catalog Zones.
- * Proof of concept python scripts (<https://github.com/IETF-Hackathon/NSDCatZ>) that can be used for both generating and consuming DNS Catalog Zones with NSD have been developed during the hackathon at the IETF-109.

Interoperability between the above implementations has been tested during the hackathon at the IETF-109.

Appendix B. Change History (to be removed before final publication)

- * draft-muks-dnsop-dns-catalog-zones-00
 - | Initial public draft.
- * draft-muks-dnsop-dns-catalog-zones-01
 - | Added Witold, Ray as authors. Fixed typos, consistency issues. Fixed references. Updated Area. Removed newly introduced custom RR TYPES. Changed schema version to 1. Changed TSIG requirement from MUST to SHOULD. Removed restrictive language about use of DNS QUERY. When zones are introduced into a catalog zone, a primary SHOULD first make the new zones available for transfers first (instead of MUST). Updated examples, esp. use IPv6 in examples per Fred Baker. Add catalog zone example.
- * draft-muks-dnsop-dns-catalog-zones-02
 - | Addressed some review comments by Patrik Lundin.
- * draft-muks-dnsop-dns-catalog-zones-03
 - | Revision bump.

* draft-muks-dnsop-dns-catalog-zones-04

Reordering of sections into more logical order. Separation of multi-valued properties into their own category.

* draft-toorop-dnsop-dns-catalog-zones-00

New authors to pickup the editor pen on this draft

Remove data type definitions for zone properties Removing configuration of member zones through zone properties altogether

Remove Open issues and discussion Appendix, which was about zone options (including primary/secondary relationships) only.

* draft-toorop-dnsop-dns-catalog-zones-01

Added a new section "The Serial Property", introducing a new mechanism which can help with disseminating zones from the primary to the secondary nameservers in a timely fashion more reliably.

Three different ways to provide a "serial" property with a member zone are offered to or the workgroup for discussion.

Added a new section "Implementation Status", listing production ready, upcoming and Proof of Concept implementations, and reporting on interoperability of the different implementations.

* draft-toorop-dnsop-dns-catalog-zones-02

Adding the coo property for zone migration in a controlled fashion

Adding the group property for reconfigure settings of member zones in an atomic update

Adding the epoch property to reset zone associated state in a controlled fashion

* draft-toorop-dnsop-dns-catalog-zones-03

Big cleanup!

Introducing the terms catalog consumer and catalog producer

Reorganized topics to create a more coherent whole

Properties all have consistent format now

Try to assume the least possible from implementations w.r.t.:

- 1) Predictability of the <unique-N> IDs of member zones
- 2) Whether or not fallback catalog zones can be found for a member
- 3) Whether or not a catalog consumer can maintain state

* draft-toorop-dnsop-dns-catalog-zones-04

Move Implementation status to appendix

Miscellaneous textual improvements

coo property points to \$NEWCATZ (and not zones.\$NEWCATZ)

Remove suggestion to increase serial and remove member zone from \$OLDCATZ after migration

More consistent usage of the terms catalog consumer and catalog producer throughout the document

Better (safer) description of resetting refresh timers of member zones with the serial property

Removing a member MUST remove zone associated state

Make authentication requirements a bit less prescriptive in security considerations

Updated implementation status for KnotDNS

Describe member node name changes and update "Zone associated state reset" to use that as the mechanism for it.

Add Peter Thomassen as co-author

Complete removal of the epoch property. We consider consumer optimizations with predictable member node labels (for example based on a hash) out of the scope of this document.

Miscellaneous editorial improvements

* draft-toorop-dnsop-dns-catalog-zones-05

Add Kees Monshouwer as co-author

Removed the "serial" property

| Allow custom properties on the global level

Authors' Addresses

Peter van Dijk
PowerDNS
Den Haag
Netherlands
Email: peter.van.dijk@powerdns.com

Libor Peltan
CZ.NIC
Czechia
Email: libor.peltan@nic.cz

Ondrej Sury
Internet Systems Consortium
Czechia
Email: ondrej@isc.org

Willem Toorop
NLnet Labs
Science Park 400
1098 XH Amsterdam
Netherlands
Email: willem@nlnetlabs.nl

Kees Monshouwer
Netherlands
Email: mind@monshouwer.eu

Peter Thomassen
deSEC, SSE - Secure Systems Engineering
Berlin
Germany
Email: peter@desec.io

Network Working Group
Internet-Draft
Updates: 5155, 6014, 8624 (if approved)
Intended status: Standards Track
Expires: 10 April 2022

P. Hoffman
ICANN
7 October 2021

Revised IANA Considerations for DNSSEC
draft-ietf-dnsop-dnssec-iana-cons-05

Abstract

This document changes the review requirements needed to get DNSSEC algorithms and resource records added to IANA registries. It updates RFC 6014 to include hash algorithms for DS (Delegation Signer) records and NSEC3 (Hashed Authenticated Denial of Existence) parameters. It also updates RFC 5155 and RFC 6014, which have requirements for DNSSEC algorithms, and updates RFC 8624 to say that algorithms that are described in RFCs that are not on standards track are only at the "MAY" level of implementation recommendation. The rationale for these changes is to bring the requirements for DS records and for the hash algorithms used in NSEC3 in line with the requirements for all other DNSSEC algorithms.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 10 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Update to RFC 6014	3
3. Update to RFC 8624	3
4. IANA Considerations	3
5. Security Considerations	4
6. References	4
6.1. Normative References	4
6.2. Informative References	5
Appendix A. Acknowledgements	5
Author's Address	5

1. Introduction

DNSSEC is primarily described in [RFC4033], [RFC4034], and [RFC4035]. DNSSEC commonly uses another resource record beyond those defined in RFC 4034: NSEC3 [RFC5155]. DS resource records were originally defined in [RFC3658], and that definition was obsoleted by RFC 4034.

[RFC6014] updated the requirements for how DNSSEC cryptographic algorithm identifiers in the IANA registries are assigned, reducing the requirements from being "Standards Action" to "RFC Required". However, the IANA registry requirements for hash algorithms for DS records [RFC3658] and for the hash algorithms used in NSEC3 records [RFC5155] are still "Standards Action". This document updates those IANA registry requirements. (For reference on how IANA registries can be updated in general, see [RFC8126].)

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Update to RFC 6014

Section 4 updates RFC 6014 to bring the requirements for DS records and NSEC3 hash algorithms in line with the rest of the DNSSEC cryptographic algorithms by allowing any DS hash algorithms, NSEC3 hash algorithms, NSEC3 parameters, and NSEC3 flags that are fully described in an RFC to have identifiers assigned in the IANA registries. This is an addition to the IANA considerations in RFC 6014.

3. Update to RFC 8624

This document updates [RFC8624] for all DNSKEY and DS algorithms that are not on standards track.

The second paragraph of Section 1.2 of RFC 8624 currently says:

This document only provides recommendations with respect to mandatory-to-implement algorithms or algorithms so weak that they cannot be recommended. Any algorithm listed in the [DNSKEY-IANA] and [DS-IANA] registries that are not mentioned in this document MAY be implemented. For clarification and consistency, an algorithm will be specified as MAY in this document only when it has been downgraded from a MUST or a RECOMMENDED to a MAY.

That paragraph is now replaced with the following:

This document provides recommendations with respect to mandatory-to-implement algorithms, algorithms so weak that they cannot be recommended, and algorithms that are defined in RFCs that are not on standards track. Any algorithm listed in the [DNSKEY-IANA] and [DS-IANA] registries that are not mentioned in this document MAY be implemented. For clarification and consistency, an algorithm will be specified as MAY in this document only when it has been downgraded from a MUST or a RECOMMENDED to a MAY.

This update is also reflected in the IANA considerations in Section 4.

4. IANA Considerations

In the "Domain Name System Security (DNSSEC) NextSECure3 (NSEC3) Parameters" registry, the registration procedure for "DNSSEC NSEC3 Flags", "DNSSEC NSEC3 Hash Algorithms", and "DNSSEC NSEC3PARAM Flags" are changed from "Standards Action" to "RFC Required".

In the "Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms" registry, the registration procedure for "Digest Algorithms" is changed from "Standards Action" to "RFC Required".

5. Security Considerations

Changing the requirements for getting security algorithms added to IANA registries as described in this document will make it easier to get good algorithms added to the registries, and will make it easier to get bad algorithms added to the registries. It is impossible to weigh the security impact of those two changes.

Administrators of DNSSEC-signed zones, and of validating resolvers, may have been making security decisions based on the contents of the IANA registries. This was a bad idea in the past, and now is an even worse idea because there will be more algorithms in those registries that may not have gone through IETF review. Security decisions about which algorithms are safe and not safe should be made by reading the security literature, not by looking in IANA registries.

6. References

6.1. 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, <<https://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, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.

- [RFC6014] Hoffman, P., "Cryptographic Algorithm Identifier Allocation for DNSSEC", RFC 6014, DOI 10.17487/RFC6014, November 2010, <<https://www.rfc-editor.org/info/rfc6014>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8624] Wouters, P. and O. Sury, "Algorithm Implementation Requirements and Usage Guidance for DNSSEC", RFC 8624, DOI 10.17487/RFC8624, June 2019, <<https://www.rfc-editor.org/info/rfc8624>>.

6.2. Informative References

- [RFC3658] Gudmundsson, O., "Delegation Signer (DS) Resource Record (RR)", RFC 3658, DOI 10.17487/RFC3658, December 2003, <<https://www.rfc-editor.org/info/rfc3658>>.

Appendix A. Acknowledgements

Donald Eastlake, Murray Kucherawy, Dan Harkins, Martin Duke, and Benjamin Kaduk contributed to this document.

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

OPSAWG Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 28 September 2022

M. Richardson
Sandelman Software Works
W. Pan
Huawei Technologies
27 March 2022

Operational Considerations for use of DNS in IoT devices
draft-ietf-opsawg-mud-iot-dns-considerations-04

Abstract

This document details concerns about how Internet of Things devices use IP addresses and DNS names. The issue becomes acute as network operators begin deploying RFC8520 Manufacturer Usage Description (MUD) definitions to control device access.

This document makes recommendations on when and how to use DNS names in MUD files.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-opsawg-mud-iot-dns-considerations/>.

Discussion of this document takes place on the opsawg Working Group mailing list (<mailto:opsawg@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/opsawg/>.

Source for this draft and an issue tracker can be found at <https://github.com/mcr/iot-mud-dns-considerations>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Strategies to map names	4
4. DNS and IP Anti-Patterns for IoT device Manufacturers	6
4.1. Use of IP address literals in-protocol	7
4.2. Use of non-deterministic DNS names in-protocol	8
4.3. Use of a too inclusive DNS name	8
5. DNS privacy and outsourcing versus MUD controllers	9
6. Recommendations to IoT device manufacturer on MUD and DNS usage	9
6.1. Consistently use DNS	10
6.2. Use primary DNS names controlled by the manufacturer	10
6.3. Use Content-Distribution Network with stable names	10
6.4. Do not use geofenced names	10
6.5. Prefer DNS servers learnt from DHCP/Route Advertisements	10
7. Privacy Considerations	11
8. Security Considerations	12
9. References	12
9.1. Normative References	13
9.2. Informative References	14
Appendix A. Appendices	15
Authors' Addresses	15

1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources. Access Control Lists (ACLs) can be defined in an RFC8520 Manufacturer Usage Description (MUD) file that permit a device to access Internet resources by DNS name.

Use of a DNS name rather than IP address in the ACL has many advantages: not only does the layer of indirection permit the mapping of name to IP address to be changed over time, it also generalizes automatically to IPv4 and IPv6 addresses, as well as permitting load balancing of traffic by many different common ways, including multi-CDN deployments wherein load balancing can account for geography and load.

At the MUD policy enforcement point -- the firewall -- there is a problem. The firewall has only access to the layer-3 headers of the packet. This includes the source and destination IP address, and if not encrypted by IPsec, the destination UDP or TCP port number present in the transport header. The DNS name is not present!

It has been suggested that one answer to this problem is to provide a forced intermediate for the TLS connections. This could in theory be done for TLS 1.2 connections. The MUD policy enforcement point could observe the Server Name Identifier (SNI) [RFC6066]. Some Enterprises do this already. But, as this involves active termination of the TCP connection (a forced circuit proxy) in order to see enough of the traffic, it requires significant effort. But, TLS 1.3 provides options to encrypt the SNI as the ESNI, which renders the practice useless in the end.

So in order to implement these name based ACLs, there must be a mapping between the names in the ACLs and layer-3 IP addresses. The first section of this document details a few strategies that are used.

The second section of this document details how common manufacturer anti-patterns get in the way this mapping.

The third section of this document details how current trends in DNS resolution such as public DNS servers, DNS over TLS (DoT), and DNS over HTTPS (DoH) cause problems for the strategies employed. Poor interactions with content-distribution networks is a frequent pathology that can result.

The fourth section of this document makes a series of recommendations ("best current practices") for manufacturers on how to use DNS, and IP addresses with specific purpose IoT devices.

The Privacy Considerations section concerns itself with issues that DNS-over-TLS and DNS-over-HTTPS are frequently used to deal with. How these concerns apply to IoT devices located within a residence or enterprise is a key concern.

The Security Considerations section covers some of the negative outcomes should MUD/firewall managers and IoT manufacturers choose not to cooperate.

2. Terminology

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Strategies to map names

The most naive method is to try to map IP addresses to names using the in-addr.arpa (IPv4), and ipv6.arpa (IPv6) mappings. This fails for a number of reasons:

1. it can not be done fast enough,
2. it reveals usage patterns of the devices,
3. the mapping are often incomplete,
4. even if the mapping is present, due to virtual hosting, it may not map back to the name used in the ACL.

This is not a successful strategy, its use is NOT RECOMMENDED.

XXX --- explain in detail how this can fail.

XXX --- explain N:1 vs 1:1 for virtual hosting.

The simplest successful strategy for translating names is for a MUD controller to take is to do a DNS lookup on the name (a forward lookup), and then use the resulting IP addresses to populate the physical ACLs.

There are still a number of failures possible.

The most important one is in the mapping of the names to IP addresses may be non-deterministic. [RFC1794] describes the very common mechanism that returns DNS A (or reasonably AAAA) records in a permuted order. This is known as Round Robin DNS, and it has been used for many decades. The device is intended to use the first IP address that is returned, and each query returns addresses in a different ordering, splitting the load among many servers.

This situation does not result in failures as long as all possible A/AAAA records are returned. The MUD controller and the device get a matching set, and the ACLs that are setup cover all possibilities.

There are a number of circumstances in which the list is not exhaustive. The simplest is when the round robin does not return all addresses. This is routinely done by geographical DNS load balancing system. It can also happen if there are more addresses than will conveniently fit into a DNS reply. The reply will be marked as truncated. (If DNSSEC resolution will be done, then the entire RR must be retrieved over TCP (or using a larger EDNS(0) size) before being validated)

However, in a geographical DNS load balancing system, different answers are given based upon the locality of the system asking. There may also be further layers of round-robin indirection.

Aside from the list of records being incomplete, the list may have changed between the time that the MUD controller did the lookup and the time that the IoT device does the lookup, and this change can result in a failure for the ACL to match.

In order to compensate for this, the MUD controller SHOULD regularly do DNS lookups in order to get never have stale data. These lookups need to be rate limited in order to avoid load. It may be necessary to avoid local recursive DNS servers. The MUD controller SHOULD incorporate its own recursive caching DNS server. Properly designed recursive servers should cache data for many minutes to days, while the underlying DNS data can change at a higher frequency, providing different answers to different queries!

A MUD controller that is aware of which recursive DNS server that the IoT device will use can instead query that server on a periodic basis. Doing so provides three advantages:

1. any geographic load balancing will base the decision on the geolocation of the recursive DNS server, and the recursive name server will provide the same answer to the MUD controller as to the IoT device.
2. the resulting name to IP address mapping in the recursive name server will be cached, and will remain the same for the entire advertised Time-To-Live reported in the DNS query return. This also allows the MUD controller to avoid doing unnecessary queries.
3. if any addresses have been omitted in a round-robin DNS process, the cache will have the set of addresses that were returned.

The solution of using the same caching recursive resolver as the target device is very simple when the MUD controllers is located in a residential CPE device. The device is usually also the policy enforcement point for the ACLs, and a caching resolver is typically located on the same device. In addition the convenience, there is a shared fate advantage: as all three components are running on the same device, if the device is rebooted, clearing the cache, then all three components will get restarted when the device is restarted.

Where the solution is more complex is when the MUD controller is located elsewhere in an Enterprise, or remotely in a cloud such as when a Software Defines Network (SDN) is used to manage the ACLs. The DNS servers for a particular device may not be known to the MUD controller, nor the MUD controller be even permitted to make recursive queries that server if it is known. In this case, additional installation specific mechanisms are probably needed to get the right view of DNS.

4. DNS and IP Anti-Patterns for IoT device Manufacturers

In many design fields, there are good patterns that should be emulated, and often there are patterns that should not be emulated. The latter are called anti-patterns, as per [antipatterns].

This section describes a number of things with IoT manufacturers have been observed to do in the field, each of which presents difficulties for MUD enforcement points.

4.1. Use of IP address literals in-protocol

A common pattern for a number of devices is to look for firmware updates in a two step process. An initial query is made (often over HTTPS, sometimes with a POST, but the method is immaterial) to a vendor system that knows whether or not an update is required.

The current firmware model of the device is sometimes provided and then the authoritative server provides a determination if a new version is required, and if so, what version. In simpler cases, an HTTPS end point is queried which provides the name and URL of the most recent firmware.

The authoritative upgrade server then responds with a URL of a firmware blob that the device should download and install. Best practice is that firmware is either signed internally ([I-D.ietf-suit-architecture]) so that it can be verified, or a hash of the blob is provided.

An authoritative server might be tempted to provide an IP address literal inside the protocol: there are two arguments (anti-patterns) for doing this.

One is that it eliminates problems to firmware updates that might be caused by lack of DNS, or incompatibilities with DNS. For instance the bug that causes interoperability issues with some recursive servers would become unpatchable for devices that were forced to use that recursive resolver type.

A second reason to avoid a DNS in the URL is when an inhouse content-distribution system is involved that involves on-demand instances being added (or removed) from a cloud computing architecture.

But, there are many problems with use of IP address literals for the location of the firmware.

The first is that the update service server must decide whether to provide an IPv4 or an IPv6 literal. A DNS name can contain both kinds of addresses, and can also contain many different IP addresses of each kind.

The second problem is that it forces the MUD file definition to contain the exact same IP address literals. It must also contain an ACL for each address literal. DNS provides a useful indirection method that naturally aggregates the addresses.

A third problem involves the use of HTTPS. IP address literals do not provide enough context for TLS ServerNameIndicator to be useful [RFC6066]. This limits the firmware repository to be a single tenant on that IP address, and for IPv4 (at least), this is no longer a sustainable use of IP addresses.

And with any non-deterministic name or address that is returned, the MUD controller is not challenged to validate the transaction, as it can not see into the communication.

Third-party content-distribution networks (CDN) tend to use DNS names in order to isolate the content-owner from changes to the distribution network.

4.2. Use of non-deterministic DNS names in-protocol

A second pattern is for a control protocol to connect to a known HTTP end point. This is easily described in MUD. Within that control protocol references are made to additional content at other URLs. The values of those URLs do not fit any easily described pattern and may point at arbitrary names.

Those names are often within some third-party Content-Distribution-Network (CDN) system, or may be arbitrary names in a cloud-provider storage system such as Amazon S3 (such [AmazonS3], or [Akamai]).

Such names may be unpredictably chosen by the content provider, and not the content owner, and so impossible to insert into a MUD file.

Even if the content provider chosen names are deterministic they may change at a rate much faster than MUD files can be updated.

This in particular may apply to the location where firmware updates may be retrieved.

4.3. Use of a too inclusive DNS name

Some CDNs make all customer content at a single URL (such as s3.amazonaws.com). This seems to be ideal from a MUD point of view: a completely predictable URL.

The problem is that a compromised device could then connect to the contents of any bucket, potentially attacking the data from other customers.

Exactly what the risk is depends upon what the other customers are doing: it could be limited to simply causing a distributed denial of service attack resulting to high costs to those customers, or such an attack could potentially include writing content.

Amazon has recognized the problems associated with this practice, and aims to change it to a virtual hosting model, as per [awss3virtualhosting].

The MUD ACLs provide only for permitting end points (hostnames and ports), but do not filter URLs (nor could filtering be enforced within HTTPS).

5. DNS privacy and outsourcing versus MUD controllers

[RFC7858] and [RFC8094] provide for DNS over TLS (DoT) and DNS over HTTPS (DoH). [I-D.ietf-dnsop-terminology-ter] details the terms. But, even with traditional DNS over Port-53 (Do53), it is possible to outsource DNS queries to other public services, such as those operated by Google, CloudFlare, Verisign, etc.

For some users and classes of device, revealing the DNS queries to those outside entities may constitute a privacy concern. For other users the use of an insecure local resolver may constitute a privacy concern.

As described above in Section 3 the MUD controller needs to have access to the same resolver(s) as the IoT device.

6. Recommendations to IoT device manufacturer on MUD and DNS usage

Inclusion of a MUD file with IoT devices is operationally quite simple. It requires only a few small changes to the DHCP client code to express the MUD URL. It can even be done without code changes via the use of a QR code affixed to the packaging (see [I-D.richardson-mud-qrcode]).

The difficult part is determining what to put into the MUD file itself. There are currently tools that help with the definition and analysis of MUD files, see [mudmaker]. The remaining difficulty is now the semantic contents of what is in the MUD file. An IoT manufacturer must now spend some time reviewing what the network communications that their device does.

This document has discussed a number of challenges that occur relating to how DNS requests are made and resolved, and it is the goal of this section to make recommendations on how to modify IoT systems to work well with MUD.

6.1. Consistently use DNS

For the reasons explained in Section 4.1, the most important recommendation is to avoid using IP address literals in any protocol. Names should always be used.

6.2. Use primary DNS names controlled by the manufacturer

The second recommendation is to allocate and use names within zones controlled by the manufacturer. These names can be populated with an alias (see [RFC8499] section 2) that points to the production system. Ideally, a different name is used for each logical function, allowing for different rules in the MUD file to be enabled and disabled.

While it used to be costly to have a large number of aliases in a web server certificate, this is no longer the case. Wildcard certificates are also commonly available which allowed for an infinite number of possible names.

6.3. Use Content-Distribution Network with stable names

When aliases point to a Content-Distribution Network (CDN), prefer to use stable names that point to appropriately load balanced targets. CDNs that employ very low time-to-live (TTL) values for DNS make it harder for the MUD controller to get the same answer as the IoT Device. A CDN that always returns the same set of A and AAAA records, but permutes them to provide the best one first provides a more reliable answer.

6.4. Do not use geofenced names

Due the problems with different answers from different DNS servers, described above, a strong recommendation is to avoid using such things.

6.5. Prefer DNS servers learnt from DHCP/Route Advertisements

IoT Devices should prefer doing DNS to the network provided DNS servers. Whether this is restricted to Classic DNS (Do53) or also includes using DoT/DoH is a local decision, but a locally provided DoT server SHOULD be used, as recommended by [I-D.reddy-dprive-bootstrap-dns-server] and [I-D.peterson-doh-dhcp].

The ADD WG is currently only focusing on insecure discovery mechanisms like DHCP/RA [I-D.btw-add-home] and DNS based discovery mechanisms ({I-D.pauly-add-deer}). Secure discovery of network provided DoH/DoT resolver is possible using the mechanisms discussed in [I-D.reddy-add-enterprise] section-4.

Use of public QuadX resolver instead of the provided DNS resolver, whether Do53, DoT or DoH is discouraged. Should the network provide such a resolver for use, then there is no reason not to use it, as the network operator has clearly thought about this.

Some manufacturers would like to have a fallback to using a public resolver to mitigate against local misconfiguration. There are a number of reasons to avoid this, or at least do this very carefully.

It is recommended that use of non-local resolvers is only done when the locally provided resolvers provide no answers to any queries at all, and do so repeatedly. The use of the operator provided resolvers SHOULD be retried on a periodic basis, and once they answer, there should be no further attempts to contact public resolvers.

Finally, the list of public resolvers that might be contacted MUST be listed in the MUD file as destinations that are to be permitted! This should include the port numbers (53, 853 for DoT, 443 for DoH) that will be used as well.

7. Privacy Considerations

The use of non-local DNS servers exposes the list of names resolved to a third parties, including passive eavesdroppers.

The use of DoT and DoH eliminates the minimizes threat from passive eavesdropped, but still exposes the list to the operator of the DoT or DoH server. There are additional methods, such as described by [I-D.pauly-dprive-oblivious-doh].

The use of unencrypted (Do53) requests to a local DNS server exposes the list to any internal passive eavesdroppers, and for some situations that may be significant, particularly if unencrypted WiFi is used. Use of Encrypted DNS connection to a local DNS recursive resolver is a preferred choice, assuming that the trust anchor for the local DNS server can be obtained, such as via [I-D.reddy-dprive-bootstrap-dns-server].

IoT devices that reach out to the manufacturer at regular intervals to check for firmware updates are informing passive eavesdroppers of the existence of a specific manufacturer's device being present at the origin location.

Identifying the IoT device type empowers the attacker to launch targeted attacks to the IoT device (e.g., Attacker can advantage of the device vulnerability).

While possession of a Large (Kitchen) Appliance at a residence may be uninteresting to most, possession of intimate personal devices (e.g., "sex toys") may be a cause for embarrassment.

IoT device manufacturers are encouraged to find ways to anonymize their update queries. For instance, contracting out the update notification service to a third party that deals with a large variety of devices would provide a level of defense against passive eavesdropping. Other update mechanisms should be investigated, including use of DNSSEC signed TXT records with current version information. This would permit DoT or DoH to convey the update notification in a private fashion. This is particularly powerful if a local recursive DoT server is used, which then communicates using DoT over the Internet.

The more complex case of section Section 4.1 postulates that the version number needs to be provided to an intelligent agent that can decide the correct route to do upgrades. The current [I-D.ietf-suit-architecture] specification provides a wide variety of ways to accomplish the same thing without having to divulge the current version number.

The use of a publically specified firmware update protocol would also enhance privacy of IoT devices. In such a system the IoT device would never contact the manufacturer for version information or for firmware itself. Instead, details of how to query and where to get the firmware would be provided as a MUD extension, and a Enterprise-wide mechanism would retrieve firmware, and then distribute it internally. Aside from the bandwidth savings of downloading the firmware only once, this also makes the number of devices active confidential, and provides some evidence about which devices have been upgraded and which ones might still be vulnerable. (The unpatched devices might be lurking, powered off, lost in a closet)

8. Security Considerations

This document deals with conflicting Security requirements:

1. devices which an operator wants to manage using [RFC8520]
2. requirements for the devices to get access to network resources that may be critical to their continued safe operation.

This document takes the view that the two requirements do not need to be in conflict, but resolving the conflict requires some advance planning by all parties.

9. References

9.1. Normative References

- [Akamai] "Akamai", 2019,
<https://en.wikipedia.org/wiki/Akamai_Technologies>.
- [AmazonS3] "Amazon S3", 2019,
<https://en.wikipedia.org/wiki/Amazon_S3>.
- [I-D.ietf-dnsop-terminology-ter]
Hoffman, P., "Terminology for DNS Transports and Location", Work in Progress, Internet-Draft, draft-ietf-dnsop-terminology-ter-02, 3 August 2020,
<<https://www.ietf.org/archive/id/draft-ietf-dnsop-terminology-ter-02.txt>>.
- [I-D.ietf-suit-architecture]
Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", Work in Progress, Internet-Draft, draft-ietf-suit-architecture-16, 27 January 2021, <<https://www.ietf.org/archive/id/draft-ietf-suit-architecture-16.txt>>.
- [I-D.peterson-doh-dhcp]
Peterson, T., "DNS over HTTP resolver announcement Using DHCP or Router Advertisements", Work in Progress, Internet-Draft, draft-peterson-doh-dhcp-01, 21 October 2019, <<https://www.ietf.org/archive/id/draft-peterson-doh-dhcp-01.txt>>.
- [I-D.reddy-dprive-bootstrap-dns-server]
Reddy, T., Wing, D., Richardson, M. C., and M. Boucadair, "A Bootstrapping Procedure to Discover and Authenticate DNS-over-TLS and DNS-over-HTTPS Servers", Work in Progress, Internet-Draft, draft-reddy-dprive-bootstrap-dns-server-08, 6 March 2020,
<<https://www.ietf.org/archive/id/draft-reddy-dprive-bootstrap-dns-server-08.txt>>.
- [RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794, DOI 10.17487/RFC1794, April 1995,
<<https://www.rfc-editor.org/info/rfc1794>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

9.2. Informative References

- [antipatterns] "AntiPattern", 12 July 2021, <<https://www.agilealliance.org/glossary/antipattern>>.
- [awss3virtualhosting] "Down to the Wire: AWS Delays 'Path-Style' S3 Deprecation at Last Minute", 12 July 2021, <<https://techmonitor.ai/technology/cloud/aws-s3-path-deprecation>>.
- [I-D.btw-add-home] Boucadair, M., Reddy, T., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for Encrypted DNS Discovery", Work in Progress, Internet-Draft, draft-btw-add-home-12, 22 January 2021, <<https://www.ietf.org/archive/id/draft-btw-add-home-12.txt>>.

[I-D.pauly-dprive-oblivious-doh]

Kinnear, E., McManus, P., Pauly, T., Verma, T., and C. A. Wood, "Oblivious DNS Over HTTPS", Work in Progress, Internet-Draft, draft-pauly-dprive-oblivious-doh-11, 17 February 2022, <<https://www.ietf.org/archive/id/draft-pauly-dprive-oblivious-doh-11.txt>>.

[I-D.reddy-add-enterprise]

Reddy, T. and D. Wing, "DNS-over-HTTPS and DNS-over-TLS Server Deployment Considerations for Enterprise Networks", Work in Progress, Internet-Draft, draft-reddy-add-enterprise-00, 23 June 2020, <<https://www.ietf.org/archive/id/draft-reddy-add-enterprise-00.txt>>.

[I-D.richardson-mud-qrcode]

Richardson, M., Latour, J., and H. H. Gharakheili, "On loading MUD URLs from QR codes", Work in Progress, Internet-Draft, draft-richardson-mud-qrcode-07, 21 March 2022, <<https://www.ietf.org/archive/id/draft-richardson-mud-qrcode-07.txt>>.

[mudmaker] "Mud Maker", 2019, <<https://mudmaker.org>>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.

Appendix A. Appendices

Authors' Addresses

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

DNSOP WG
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2021

T. Reddy
McAfee
N. Cook
Open-Xchange
D. Wing
Citrix
M. Boucadair
Orange
June 4, 2021

DNS Access Denied Error Page
draft-reddy-dnsop-error-page-08

Abstract

When a DNS server filters a query, the response to such query conveys no detailed explanation that elaborates why that query was blocked, leading thus to end-user confusion. A solution to this problem is needed in order to enhance the user experience.

This document defines a method to return an URI that explains the reason why a DNS query was filtered by a DNS server.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 6, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Terminology	6
3. Error Page URI EDNS0 Option Format	6
4. Error Page URI Processing	7
4.1. Mitigating EDNS0 Forgery	8
5. Error Page	9
6. Usability Considerations	10
7. Security Considerations	10
8. IANA Considerations	11
8.1. A New Error Page URI EDNS Option	11
9. Acknowledgements	11
10. References	11
10.1. Normative References	11
10.2. Informative References	13
Authors' Addresses	14

1. Introduction

DNS filters are deployed for a variety of reasons, including endpoint security, parental filtering, and filtering required by law enforcement. Some of these reasons are discussed in more detail below:

- o Various network security services are provided by Enterprise networks to protect endpoints (e.g., Hosts including IoT devices). Network-based security solutions such as firewalls and Intrusion Prevention Systems (IPS) rely upon network traffic inspection to implement perimeter-based security policies. The network security services may, for example, prevent malware download, block known malicious domains, block phishing sites, etc.

These network security services act on DNS queries originating from endpoints. For example, DNS firewalls, a method of expressing DNS response policy information inside specially constructed DNS zones, known as Response Policy Zones (RPZs) allows DNS servers to modify their DNS responses in real time in order to stop access to malware and phishing domains. Note that

some of the commonly known types of malware are viruses, worms, trojans, bots, ransomware, backdoors, spyware, and adware.

- o Network devices in a home network offer network security to protect the devices within the home network by performing DNS-based content filtering. The network security service may, for example, block access to specific domains to enforce parental control, block access to malware sites, etc.
- o Internet Service Providers (ISPs) typically block access to some DNS domains due to a requirement imposed by an external entity (e.g., Law Enforcement Agency). Such blocking is performed using DNS-based content filtering.

DNS responses can be filtered by sending a bogus (also called, "forged") A or AAAA response, NXDOMAIN error or empty answer, or an extended DNS error (EDE) code defined in [RFC8914]. Each of these methods have advantages and disadvantages that are discussed below:

1. The DNS response is forged to provide a list of IP addresses that points to an HTTP(S) server alerting the end user about the reason for blocking access to the requested domain (e.g., malware). When an HTTP(S) enabled domain name is blocked, the network security device (e.g., CPE, firewall) presents a block page instead of the HTTP response from the content provider hosting that domain. If an HTTP enabled domain name is blocked, the network security device intercepts the HTTP request and returns a block page over HTTP. If an HTTPS enabled domain is blocked, the block page is also served over HTTPS. In order to return a block page over HTTPS, man in the middle (MITM) is enabled on endpoints by generating a local root certificate and an accompanying (local) public/private key pair. The local root certificate is installed on the endpoint while the network security device(s) stores a copy of the private key. During the TLS handshake, the network security device modifies the certificate provided by the server and (re)signs it using the private key from the local root certificate.
 - * However, configuring the local root certificate on endpoints is not a viable option in several deployments like home networks, schools, Small Office/Home Office (SOHO), and Small/Medium Enterprise (SME). In these cases, the typical behavior is that the forged DNS response directs the user towards a server hosted to display the block page which breaks the TLS connection. For web-browsing this then results in an HTTPS certificate error message indicating that a secure connection could not be established, which gives no information to the end-user about the reason for the error.

The typical errors are "The security certificate presented by this website was not issued by a trusted certificate authority" (Internet Explorer/Edge), "The site's security certificate is not trusted" (Chrome), "This Connection is Untrusted" (Firefox), "Safari can't verify the identity of the website..." (Safari on MacOS)".

- * Enterprise networks do not assume that all the connected devices are managed by the IT team or Mobile Device Management (MDM) devices, especially in the quite common Bring Your Own Device (BYOD) scenario. In addition, the local root certificate cannot be installed on IoT devices without a device management tool.
 - * An end user does not know why the connection was reset and, consequently, may repeatedly try to reach the domain but with no success. Frustrated, the end user may switch to an alternate network that offers no DNS-level protection against malware and phishing, potentially compromising both security and privacy. Furthermore, certificate errors train users to click through certificate errors, which is a bad security practice. To eliminate the need for an end user to click through certificate errors, an end user may manually install a local root certificate on a host device (e.g. [Chrome-Install-Cert]). Doing so, however, is also a bad security practice as it creates a security vulnerability that may be exploited by a MITM attack. When a manually installed local root certificate expires, the user has to (again) manually install the new local root certificate.
2. The DNS response is forged to provide a NXDOMAIN response to cause the DNS lookup to terminate in failure. In this case, an end user does not know why the domain cannot be reached and may repeatedly try to reach the domain but with no success. Frustrated, the end user may use insecure connections to reach the domain, potentially compromising both security and privacy.
 3. The extended error codes Blocked, Censored, and Filtered defined in Section 4 of [RFC8914] can be returned by a DNS server to provide additional information about the cause of an DNS error. If the extended error code "Forged Answer" defined in Section 4.5 of [RFC8914] is returned by the DNS server, the client can identify the DNS response is forged together with the reason for HTTPS certificate error.

These extended error codes do not suffer from the limitations discussed in bullets (1) and (2), but the user still does not know the exact reason nor he/she is aware of the exact entity

blocking the access to the domain. For example, a DNS server may block access to a domain based on the content category such as "Adult Content" to enforce parental control, "Violence & Terrorism" due to an external requirement imposed by an external entity (e.g., Law Enforcement Agency), etc. These content categories cannot be standardized because the classification of domains into content categories is vendor specific, typically ranges from 40 to 100 types of categories depending on the vendor and the categories keep evolving. Furthermore, the threat data used to categorize domains may sometimes misclassify domains (e.g., domains wrongly classified as Domain Generation Algorithm (DGA) by deep learning techniques, domain wrongly classified as phishing due to crowd sourcing, new domains not categorized by the threat data). A user needs to know the contact details of the IT/InfoSec team to raise a complaint.

4. The EXTRA-TEXT field of the EDE option defined in Section 2 of [RFC8914] can include additional textual information about the cause of the error, but the information could be provided in a language that is not understood by the user. When a resolver or forwarder forwards the received EDE option, the EXTRA-TEXT field only conveys the source of the error (Section 3 of [RFC8914]) and does not provide additional textual information about the cause of the error. Most importantly, EDE option does not offer authenticated information; it can thus be spoofed by an attacker. In addition, the additional textual information may not be able to convey all of the required information about the cause of the DNS error because lengthy EXTRA-TEXT content would be truncated to prevent fragmentation (Section 3 of [RFC8914]).

No matter which type of response is generated (forged IP address(es), NXDOMAIN or empty answer, or an extended error code), the user who triggered the DNS query has little chance to understand which entity filtered the query, how to report a mistake in the filter, or why the entity filtered it at all. This document describes a mechanism to provide an URI which, when accessed, provides such information to the user.

One of the other benefits of this approach is to eliminate the need to "spoof" block pages for HTTPS resources. This is achieved as the block page no longer needs to create a signed certificate when blocking a destination. This approach avoids the need to install a local root certificate authority on those IT-managed devices.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [RFC8499].

'Encrypted DNS' refers to any encrypted scheme to convey DNS messages, for example, DNS-over-HTTPS [RFC8484], DNS-over-TLS [RFC7858], or DNS-over-QUIC [I-D.ietf-dprive-dnsquic].

3. Error Page URI EDNS0 Option Format

This document uses an EDNS0 [RFC6891] option to include the URI that provides additional information in a DNS response about the cause of blocking access to a requested domain. This option is structured as depicted in Figure 1.

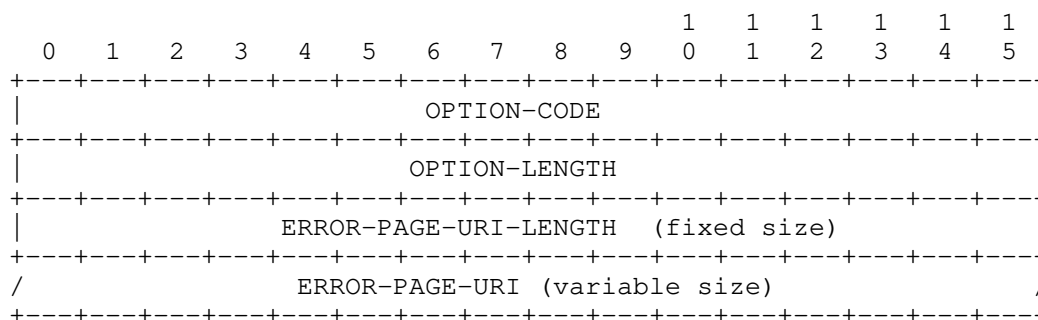


Figure 1: Error Page URI EDNS0 Option Format

The description of the fields is as follows:

- o **OPTION-CODE:** TBD, indicates the code assigned for Error Page URI (Section 6.1.2 of [RFC6891]). [RFC Editor: change TBD to the proper code once assigned by IANA.]
- o **OPTION-LENGTH:** See Section 6.1.2 of [RFC6891]. This field contains the length of the payload (everything after **OPTION-LENGTH**) in octets. The variability of the option length stems from the variable-length **ERROR-PAGE-URI** field.
- o **ERROR-PAGE-URI-LENGTH:** This 16-bit field indicates the length of **ERROR-PAGE-URI**. It **MUST NOT** be set to 0.

- o ERROR-PAGE-URI: A variable length UTF-8 encoded [RFC5198] text field containing the URI Template [RFC6570] that gives additional information about the cause of blocking access to a domain. The ERROR-PAGE-URI field MUST NOT be zero octets in length.

The Error Page URI option can be included in any response (SERVFAIL, NXDOMAIN, REFUSED, and even NOERROR, etc.) to a query that includes OPT Pseudo-RR [RFC6891].

The URI Template defined in ERROR-PAGE-URI describes how to construct the URL to fetch the error page. The agent acting as the HTTPS client on the endpoint encodes an FQDN to which access is denied into an HTTP GET request to retrieve the error page. The HTTPS server returning the error page defines the URI used by the HTTP GET request through the use of a URI Template. The URI Template is processed with a defined variable "target-domain" whose value is set to the FQDN to which access is denied.

The FQDN is provided as the variable value for "target-domain" to expand the URI Template into an URI reference in the HTTP GET request.

An example is illustrated below:

If the URI Template is "https://resolver.example.net/block-page{?target-domain}" for the HTTPS server returning the error page and access to the target domain "example.com" is blocked by the encrypted DNS server, the variable "target-domain" has the value "example.com" sent in an HTTP GET request. In the above example, the expansion of the above URI Template is "https://resolver.example.net/block-page?target-domain=example.com".

HTTP/2 [RFC7540] is the minimum RECOMMENDED HTTP version to use to retrieve the error page. The HTTPS client retrieving the error page MUST verify the entire certification path as per [RFC5280]. The HTTPS client additionally uses validation techniques described in [RFC6125] to compare the domain name in the error page URI to the server certificate provided in TLS handshake. See [RFC7525] for additional TLS recommendations.

4. Error Page URI Processing

The DNS client MUST follow the rules below to process the Error Page URI EDNS0 option:

- o If the DNS response contains more than one Error Page URI EDNS0 option, the DNS client MUST discard all Error Page URI EDNS0 options in the DNS response.
- o The Error Page URI EDNS0 option MUST be processed by the DNS client for a "Censored", "Blocked", "Filtered" or "Forged" extended error codes and MUST be ignored for any other type of extended DNS error code. When "Censored", "Blocked", "Filtered" or "Forged" extended error code is returned in conjunction with an Error Page URI EDNS0 option, any other resource records in the answer MUST be ignored by clients supporting this specification.
- o The DNS client MUST reject the error page URI if the scheme is not "https".

4.1. Mitigating EDNS0 Forgery

The Error Page URI EDNS0 option is susceptible to forgery. An attacker (e.g., a man in the middle (MITM)) could insert an extended Error Page URI EDNS0 option into the DNS response causing a client to attempt to visit that URI. For instance, the attacker can be located between the stub resolver and DNS recursive server or between the DNS proxy and the upstream resolver. To mitigate that attack, the following measures are enforced:

- o The DNS client MUST NOT process the DNS response with Error Page URI EDNS0 option unless DNS messages exchanged are cryptographically protected using encrypted DNS.
- o If a DNS client has enabled opportunistic privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client will either fallback to an encrypted connection without authenticating the DNS server provided by the local network or fallback to clear text DNS, and cannot exchange encrypted DNS messages. Both of these fallback mechanisms adversely impacts security and privacy. If the DNS client has enabled opportunistic privacy profile for DoT, the DNS client MUST ignore Error Page URI EDNS0 option in responses, but SHOULD process other parts of the response.
- o If a DNS client has enabled strict privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client requires an encrypted connection and successful authentication of the DNS server; this mitigates both passive eavesdropping and client redirection (at the expense of providing no DNS service if an encrypted, authenticated connection is not available). If the DNS client has enabled strict privacy profile for DoT, the client can process the DNS response with Error Page URI EDNS0 option. Note that the strict and opportunistic privacy profiles as defined in [RFC8310]

only applies to DoT protocol, there has been no such distinction made for DoH protocol.

- o If the DNS client determines that the encrypted DNS server does not offer DNS filtering service, it MUST reject the Error Page URI EDNS0 option. For example, the DNS client can learn whether the encrypted DNS resolver performs DNS-based content filtering or not by retrieving resolver information using the method defined in [I-D.reddy-add-resolver-info].
- o DNS forwarders (or DNS proxies) are supposed to propagate unknown EDNS0 options (Sections 4.1 and 4.4.1 of [RFC5625]), which means the Error Page URI EDNS0 option may get propagated by such a DNS server. To detect this scenario, the DNS client MUST verify the domain name in the Error Page URI matches the domain name of the encrypted DNS resolver. If this match fails, the DNS client MUST ignore Error Page URI EDNS0 option in the response, but SHOULD process other parts of the response.

5. Error Page

The following outlines the RECOMMENDED contents of an error page to assist the operator developing the error page:

- o The exact reason for blocking access to the domain. If the domain is blocked based on some threat data, the threat type associated with the blocked domain can be provided/displayed to the end user. For example, the reason can indicate the type of malware blocked like spyware and the damage it can do the security and privacy of the user.
- o The domain name blocked.
- o If query was blocked by regulation, a pointer to a regulatory text that mandates this query block.
- o The entity (or organization) blocking the access to the domain and contact details of the IT/InfoSec team to raise a complaint.
- o The blocked error page to not include Ads and dynamic content.

The content of the error page discussed above is non-normative, the above text only provides the guidelines and template for the error page and:

- o does not attempt to offer an exhaustive list for the contents of an error page.

- o it is not intended to form the basis of any legal/compliance for developing the error page.

6. Usability Considerations

The error page SHOULD be returned in the user's preferred language as expressed by the Accept-Language HTTP header.

7. Security Considerations

Security considerations in Section 6 of [RFC8914] and [RFC8624] need to be taken into consideration.

The Error Page URI EDNS0 option causes an HTTPS retrieval by the client. To prevent forgery of the Error Page URI EDNS0 option, this specification requires it only be sent only over an encrypted DNS channel with an authorized DNS server.

The client knows it is connecting to a HTTPS server returning the error page. To reduce threat surface the client can retrieve the Error Page URL using, for example, an isolated environment and take other precautions such as clearly labeling the page as untrusted or prevent user interaction with the page. Such isolation should prevent transmitting cookies, block JavaScript, block auto-fill of credentials or personal information, and be isolated from the user's normal environment.

Browsers perform some of the above restrictions when accessing captive portals (Section 5 of [RFC8910] or [Safari-Cookie]), during private browsing, or using containerization [Facebook-Container].

Note that the means to use a sandbox environment and a user interface presenting the error page are not covered in this document. By its nature, these aspects are implementation specific and best left to the application and user interface designers.

The encrypted DNS session provides transport security for the interaction between the DNS client and server, but DNSSEC signing and validation is not possible for the Error Page URI EDNS0 option returning the Error Page URI Template. However, this specification mandates the DNS client to not process DNS response with Error Page URI EDNS0 option if domain name in the Error Page URI does not match the domain name of the encrypted DNS server. The validation ensures both the servers are operated by the same entity and have the same origin (similar to the Same Origin Policy (SOP)).

By design, the object referenced by the error page URL potentially exposes additional information about the DNS resolution process that

may leak information. An example of this is the reason for blocking the access to the domain name and the entity blocking access to the domain.

8. IANA Considerations

8.1. A New Error Page URI EDNS Option

This document defines a new EDNS(0) option, entitled "Error Page URI", assigned a value of TBD from the "DNS EDNS0 Option Codes (OPT)" registry [to be removed upon publication:
[<http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11>]

Value	Name	Status	Reference
TBD	Error Page URI	Standard	[This document]

9. Acknowledgements

Thanks to Vittorio Bertola, Wes Hardaker, Ben Schwartz, Erid Orth, Viktor Dukhovni, Warren Kumari and Bob Harold for the comments.

10. References

10.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <<https://www.rfc-editor.org/info/rfc5198>>.
- [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, <<https://www.rfc-editor.org/info/rfc5280>>.

- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<https://www.rfc-editor.org/info/rfc5625>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <<https://www.rfc-editor.org/info/rfc6570>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.
- [RFC8624] Wouters, P. and O. Sury, "Algorithm Implementation Requirements and Usage Guidance for DNSSEC", RFC 8624, DOI 10.17487/RFC8624, June 2019, <<https://www.rfc-editor.org/info/rfc8624>>.

- [RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <<https://www.rfc-editor.org/info/rfc8914>>.

10.2. Informative References

- [Chrome-Install-Cert]
"How to manually install the Securlly SSL certificate in Chrome", <support.securlly.com/hc/en-us/articles/206081828-How-to-manually-install-the-Securlly-SSL-certificate-in-Chrome>.
- [Facebook-Container]
"Facebook container for Firefox", <<https://www.mozilla.org/en-US/firefox/facebookcontainer/>>.
- [I-D.ietf-dprive-dnssoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnssoquic-02 (work in progress), February 2021.
- [I-D.reddy-add-resolver-info]
Reddy, T. and M. Boucadair, "DNS Resolver Information", draft-reddy-add-resolver-info-03 (work in progress), April 2021.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8910] Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP and Router Advertisements (RAs)", RFC 8910, DOI 10.17487/RFC8910, September 2020, <<https://www.rfc-editor.org/info/rfc8910>>.

[Safari-Cookie]
"Isolated cookie store (CVE-2016-1730)",
<<https://support.apple.com/en-us/HT205732>>.

Authors' Addresses

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: kondtir@gmail.com

Neil Cook
Open-Xchange
UK

Email: neil.cook@noware.co.uk

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Domain Name System Operations (dnsop)	U. Wisser
Internet-Draft	The Swedish Internet Foundation
Intended status: Standards Track	S. Huque
Expires: 7 September 2022	Salesforce
	6 March 2022

DNSSEC automation
draft-wisser-dnssec-automation-03

Abstract

This document describes an algorithm and a protocol to automate DNSSEC Multi-Signer [RFC8901] "Multi-Signer DNSSEC Models" setup, operations and decommissioning. Using Model 2 of the Multi-Signer specification, where each operator has their own distinct KSK and ZSK sets (or CSK sets), [RFC8078] "Managing DS Records from the Parent via CDS/CDNSKEY" and [RFC7477] "Child-to-Parent Synchronization in DNS" to accomplish this.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Out-Of-Scope	3
1.2. Notation	3
1.3. Requirements Language	3
2. Use Cases	3
2.1. Maintaining a Multi-Signer group	4
2.2. Secure Nameserver Operator Transition	4
3. Automation Models	4
3.1. Centralized	4
3.2. Decentralized	4
3.3. Capabilities	5
4. Algorithms	5
4.1. Prerequisites	5
4.2. Definitions	5
4.2.1. DS Waiting Time	5
4.2.2. DNSKEY Waiting Time	6
4.2.3. NS Waiting Time	6
4.3. Setting up a new Multi-Signer group	6
4.4. A Signer joins the Multi-Signer group	6
4.5. A signer leaves the Multi-Signer group	7
4.6. A Signer performs a ZSK rollover	7
4.7. A Signer performs a CSK or KSK rollover	8
4.8. Algorithm rollover for the whole Multi-Signer group. . .	8
5. Signers with different algorithms in one Multi-Signer group	9
6. Acknowledgements	10
7. IANA Considerations	10
8. Implementation Status	10
9. Security Considerations	10
10. Normative References	10
11. Informative References	11
Appendix A. Change History	11
A.1. Change from 01 to 02	11
A.2. Change from 02 to 03	11
Authors' Addresses	12

1. Introduction

[RFC8901] describes the necessary steps and API for a Multi-Signer DNSSEC configuration. In this document we will combine [RFC8901] with [RFC8078] and [RFC7477] to define an automatable algorithm for setting up, operating and decommissioning of a Multi-Signer DNSSEC configuration.

One of the special cases of Multi-Signer DNSSEC is the secure change of DNS operator. Using Multi-Signer Model 2 the secure change of DNS operator can be accomplished.

1.1. Out-Of-Scope

In order for any Multi-Signer group to give consistent answers across all nameservers, the data contents of the zone also have to be synchronized (in addition to infrastructure records like NS, DNSKEY, CDS etc). This content synchronization is out-of-scope for this document (although there are a number of methods that can be used, such as making the the same updates to each operator using their respective APIs, using zone transfer in conjunction with "inline signing" at each operator, etc.)

1.2. Notation

Short definitions of expressions used in this document

Signer

An entity signing a zone

Multi-Signer Group

A group of signers that sign the same zone

Controller

An entity controlling the multi-signer group. Used in the decentralized model.

1.3. Requirements Language

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

2. Use Cases

2.1. Maintaining a Multi-Signer group

As described in [RFC8901] a Multi-Signer DNSSEC configuration has some challenges that can be overcome with the right infrastructure and following a number of steps for setup and operation.

In this document we describe, except for the initial trust, how the steps in the Multi-Signer DNSSEC setup can be automated.

2.2. Secure Nameserver Operator Transition

Changing the nameserver operator of a DNSSEC signed zone can be challenging. Currently the most common method is temporarily "going insecure". This is poor for security, and for users relying on the security of the zone. Furthermore, when DNSSEC is being used for application security functions like DANE [RFC6698], it is critical that the DNSSEC chain of trust remain unbroken during the transfer.

Multi-Signer DNSSEC Model 2 provides a mechanism for transitioning from one nameserver operator to another without "going insecure". A new operator joins the current operator in a temporary Multi-Signer group. Once that is accomplished and stable the old operator leaves the Multi-Signer group completing the transition.

3. Automation Models

Automation of the necessary steps can be categorized into two main models, centralized and decentralized. Both have pros and cons, and a zone operator should carefully choose the model that works best.

3.1. Centralized

In a centralized model the zone operator will run controller that executes all steps necessary and controls all signers.

A centralized controller needs to have authorized access to all signers. This can be achieved in a variety of different ways. For example will many service providers offer access through a REST API. Another possibility is access through Dynamic Update [RFC2136] with TSIG authentication.

3.2. Decentralized

In the decentralized models all signers will communicate with each other and execute the necessary steps on their instance only. For this signers need a specialized protocol to communicate configuration details that are not part of the zone data.

3.3. Capabilities

In order for any of the models to work the signer must support the following capabilities.

1. Add DNSKEY records (without the private key)
2. Remove (previously added) DNSKEY record(s)
3. Add CDS and CDNSKEY records for keys not in the DNSKEY set
4. Remove (previously added) CDS and CDNSKEY records
5. Add CSYNC record
6. Remove CSYNC record

4. Algorithms

4.1. Prerequisites

Each Signer to be added, including the initial Signer, must meet the following prerequisites before joining the Multi-Signer Group

1. A working setup of the zone, including DNSSEC signing.
2. Uses the same algorithm for DNSSEC signing as the Multi-Signer group uses or will use.
3. Signer or controller must be able to differentiate between its own keys and keys from others signers
4. Signer controller must be able to differentiate between NS records that are updated by itself and NS records that receive updates from other signers.
5. The domain must be covered by a CDS/CDNSKEY scanner and a CSYNC scanner. Otherwise updates to the parent zone have to be made manually.

4.2. Definitions

4.2.1. DS Waiting Time

Once the parent has picked up and published the new DS record set, the any further changes MUST to be delayed until the new DS set has propagated.

The minimum DS Waiting Time is the TTL of the DS RRset.

4.2.2. DNSKEY Waiting Time

Once the DNSKEY sets of all signers are updated, any further changes MUST to be delayed until the new DNSKEY set has propagated.

The minimum DNSKEY Waiting Time is the maximum of all DNSKEYS TTL values from all signers plus the time it takes to publish the zone on all secondaries.

4.2.3. NS Waiting Time

Once the parent has picked up and published the new NS record set, any further changes MUST be delayed until the new NS set has propagated.

The minimum NS Waiting Time is the maximum of the TTL value of the NS set in the parent zone and all NS sets from all signers.

4.3. Setting up a new Multi-Signer group

The zone is already authoritatively served by one DNS operator and is DNSSEC signed. For full automation both the KSK and ZSK or CSK must be online.

This would be a special case, a Multi-Signer group with only one signer.

4.4. A Signer joins the Multi-Signer group

1. Confirm that the incoming Signer meets the prerequisites.
2. Establish a trust mechanism between the Multi-Signer group and the Signer.
3. Add ZSK for each signer to all other Signers.
4. Calculate CDS/CDNSKEY Records for all KSKs/CSKs represented in the Multi-Signer group.
5. Configure all Signers with the compiled CDS/CDNSKEY RRSET.
6. Wait for Parent to publish the combined DS RRset.
7. Remove CDS/CDNSKEY Records from all Signers. (optional)
8. Wait maximum of DS-Wait-Time and DNSKEY-Wait-Time

9. Compile NS RRSET including all NS records from all Signers.
 10. Configure all Signers with the compiled NS RRSET.
 11. Compare NS RRSET of the Signers to the Parent, if there is a difference publish CSYNC record with NS and A and AAAA bit set on all signers.
 12. Wait for Parent to publish NS.
 13. Remove CSYNC record from all signers. (optional)
- 4.5. A signer leaves the Multi-Signer group
1. Remove exiting Signer's NS records from remaining Signers
 2. Compare NS RRSET of the Signers to the Parent, if there is a difference publish CSYNC record with NS and A and AAAA bit set on remaining signers.
 3. Wait for Parent to publish NS RRSET.
 4. Remove CSYNC record from all signers. (optional)
 5. Wait NS-Wait-Time
 6. Stop the exiting Signer from answering queries.
 7. Calculate CDS/CDNSKEY Records for KSKs/CSKs published by the remaining Signers.
 8. Configure remaining Signers with the compiled CDS/CDNSKEY RRSET.
 9. Remove ZSK of the exiting Signer from remaining Signers.
 10. Wait for Parent to publish the updated DS RRset.
 11. Remove CDS/CDNSKEY set from all signers. (Optional)
- 4.6. A Signer performs a ZSK rollover
1. The signer introduces the new ZSK in its own DNSKEY RRset.
 2. Update all signers with the new ZSK.
 3. Wait DNSKEY-Wait-Time
 4. Signer can start using the new ZSK.

5. When the old ZSK is not used in any signatures by the signer, the signer can remove the old ZSK from its DNSKEY RRset.
 6. Remove ZSK from DNSKEY RRset of all signers.
- 4.7. A Signer performs a CSK or KSK rollover
1. Signer publishes new CSK / KSK in its own DNSKEY RRset.
 2. In case of CSK, add CSK to DNSKEY set of all other Signers.
 3. Signer signs DNSKEY RRset with old and new CSK / KSK.
 4. Calculate new CDS/CDNSKEY RRset and publish on all signers.
 5. Wait for parent to pickup and publish new DS RR set.
 6. Wait DS-Wait-Time + DNSKEY-Wait-Time
 7. Signer removes old CSK/KSK from its DNSKEY RR set. And removes all signatures done with this key.
 8. In case of CSK, remove old CSK from DNSKEY set of all other signers.
 9. Calculate new CDS/CDNSKEY RRset and publish on all signers.
 10. Wait for parent to pickup and publish new DS RR set.
 11. Remove CDS/CDNSKEY RR sets from all signers.
- 4.8. Algorithm rollover for the whole Multi-Signer group.
1. All signers publish KSK and ZSK or CSK using the new algorithm.
 2. All signers sign all zone data with the new keys.
 3. Wait until all signers have signed all data with the new key(s).
 4. Add new ZSK of each signer to all other Signers.
 5. Calculate new CDS/CDNSKEY RRset and publish on all signers.
 6. Wait for parent to pickup and publish new DS RR set.
 7. Wait DS-Wait-Time + DNSKEY-Wait-Time

8. Removes all keys and signatures which are using the old algorithm.
 9. Calculate new CDS/CDNSKEY RRset and publish on all signers.
 10. Wait for parent to pickup and publish new DS RR set.
 11. Remove CDS/CDNSKEY RR sets from all signers.
5. Signers with different algorithms in one Multi-Signer group

Section 2.2 of [RFC4035] states that a signed zone MUST include a DNSKEY for each algorithm present in the zone's DS RRset and expected trust anchors for the zone.

A setup where different signers use different key algorithms therefore violates [RFC4035].

According to Section 5.11 of [RFC6840] validators SHOULD NOT insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work.

So a Multi-Signer setup where different signers use different key algorithms should still validate.

This could be an acceptable risk in a situation where going insecure is not desirable or impossible and name servers have to be changed between operators which only support distinct set of key algorithms.

We have to consider the following scenarios

Validator supports both algorithms

Validation should be stable through all stages of the multi-signer algorithms.

Validator supports none of the algorithms

The validator will treat the zone as unsigned. Resolution should work through all stages of the multi-signer algorithms.

Validator supports only one of the algorithms

The validator will not be able to validate the DNSKEY RR set or any data from one of the signers. So in some cases the validator will consider the zone bogus and reply with a SERVFAIL response code.

The later scenario can be mitigated, but not fully eliminated, by selecting two well supported algorithms.

6. Acknowledgements

The authors would like to thank the following for their review of this work and their valuable comments: Steve Crocker, Eric Osterweil, Roger Murray, Jonas Andersson, Peter Thomassen.

7. IANA Considerations

8. Implementation Status

One implementation of a centralized controller which supports updates through Dynamic DNS or REST API's of several vendors has been implemented by the Swedish Internet Foundation.

The code can be found as part of the Multi-Signer project on Github <https://github.com/DNSSEC-Provisioning/multi-signer-controller>

9. Security Considerations

Every step of the multi-signer algorithms has to be carefully executed at the right time and date. Any failure could resolve in the loss of resolution for the domain.

Independently of the chosen model, it is crucial that only authorized entities will be able to change the zone data. Some providers or software installation allow to make more specific configuration on the allowed changes. All extra steps to allows as little access to change zone data as possible should be taken.

If used correctly the multi-signer algorithm will strengthen the DNS security by avoiding "going insecure" at any stage of the domain life cycle.

10. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.

- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <<https://www.rfc-editor.org/info/rfc6840>>.
- [RFC7477] Hardaker, W., "Child-to-Parent Synchronization in DNS", RFC 7477, DOI 10.17487/RFC7477, March 2015, <<https://www.rfc-editor.org/info/rfc7477>>.
- [RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <<https://www.rfc-editor.org/info/rfc8078>>.

11. Informative References

- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <<https://www.rfc-editor.org/info/rfc8901>>.

Appendix A. Change History

A.1. Change from 01 to 02

1. Trying to fix wording to be more precise
2. Added algorithm for ZSK rollover
3. Added algorithm for KSK rollover
4. Added algorithm for algorithm rollover

A.2. Change from 02 to 03

1. Fix sequence of steps in the joining procedure
2. Explicit handling of CSK cases in CSK/ KSK rollover

Authors' Addresses

Ulrich Wisser
The Swedish Internet Foundation
Box 92073
SE-12007 Stockholm
Sweden
Email: ulrich@wisser.se
URI: <https://www.internetstiftelsen.se>

Shumon Huque
Salesforce
415 Mission Street, 3rd Floor
San Francisco, CA 94105
United States of America
Email: shuque@gmail.com