Internet Engineering Task Force (IETF)
Request for Comments: 8027
BCP: 207
Category: Best Current Practice
ISSN: 2070-1721

W. Hardaker
USC/ISI
O. Gudmundsson
CloudFlare
S. Krishnaswamy
Parsons

November 2016

DNSSEC Roadblock Avoidance

Abstract

This document describes problems that a Validating DNS resolver, stub-resolver, or application might run into within a non-compliant infrastructure. It outlines potential detection and mitigation techniques. The scope of the document is to create a shared approach to detect and overcome network issues that a DNSSEC software/system may face.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes problems observable during DNSSEC ([RFC4034] [RFC4035]) deployment that derive from non-compliant infrastructure. It poses potential detection and mitigation techniques.

1.1. Notation

In this document, a "Host Validator" can either be a validating stub-resolver, such as a library that an application has linked in, or a validating resolver daemon running on the same machine. It may or may not be trying to use upstream caching resolvers during its own resolution process; both cases are covered by the tests defined in this document.

The sub-variant of this is a "Validating Forwarding Resolver", which is a resolver that is configured to use upstream Resolvers when possible. A Validating Forwarding Resolver also needs to perform the tests outlined in this document before using an upstream recursive resolver.

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

1.2. Background

Deployment of DNSSEC validation is hampered by network components that make it difficult or sometimes impossible for validating resolvers to effectively obtain the DNSSEC data they need. This can occur for many different reasons including, but not limited to, the following:

- Recursive resolvers and DNS proxies [RFC5625] not being fully DNSSEC compliant

- Resolvers not being DNSSEC aware

- "Middleboxes" actively blocking, modifying, and/or restricting outbound traffic to the DNS port (53) either UDP and/or TCP

- In-path network components not allowing UDP fragments

This document talks about ways that a Host Validator can detect the state of the network it is attached to, and ways to hopefully circumvent the problems associated with the network defects it discovers. The tests described in this document may be performed on any validating resolver to detect and prevent problems. While these recommendations are mainly aimed at Host Validators, it is prudent to perform these tests from regular validating resolvers, just to make sure things work.

There are situations where a host cannot talk directly to a Resolver; the tests below cannot address how to overcome that, and inconsistent results can be seen in such cases. This can happen, for instance, when there are DNS proxies/forwarders between the user and the actual resolvers.

1.3. Implementation Experiences

Multiple lessons learned from multiple implementations led to the development of this document, including (in alphabetical order) DNSSEC-Tools' DNSSEC-Check, DNSSEC_Resolver_Check, dnssec-trigger, and FCC_Grade.

Detecting lack of support for specified Domain Name System Key (DNSKEY) algorithms and Delegation Signer (DS) digest algorithms is outside the scope of this document, but the document provides information on how to do that. See the sample test tool: <https://github.com/ogud/DNSSEC_ALG_Check>.

This document does describe compliance tests for algorithms 5, 7, and 13 with DS digest algorithms 1 and 2.

1.3.1. Test Zone Implementation

In this document, the "test.example.com" domain is used to refer to DNS records that contain test records that have known DNSSEC properties associated with them. For example, the "badsign-a.test.example.com" domain is used below to refer to a DNS A record where the signatures published for it are invalid (i.e., they are "bad signatures" that should cause a validation failure).

At the time of this publication, the "test.dnssec-tools.org" domain implements all of these test records. Thus, it may be possible to replace "test.example.com" in this document with "test.dnssec-tools.org" when performing real-world tests.

2. Goals

This document is intended to show how a Host Validator can detect the capabilities of a recursive resolver and work around any problems that could potentially affect DNSSEC resolution. This enables the Host Validator to make use of the caching functionality of the recursive resolver, which is desirable in that it decreases network traffic and improves response times.

A Host Validator has two choices: it can wait to determine that it has problems with a recursive resolver based on the results that it is getting from real-world queries issued to it or it can proactively test for problems (Section 3) to build a workaround list ahead of time (Section 5). There are pros and cons to both of these paths that are application specific, and this document does not attempt to provide guidance about whether proactive tests should or should not be used. Either way, DNSSEC roadblock avoidance techniques ought to be used when needed and if possible.

Note: Besides being useful for Host Validators, the same tests can be used for a recursive resolver to check if its upstream connections hinder DNSSEC validation.

3. Detecting DNSSEC Non-compliance

This section outlines tests that a validator should perform in order to test certain features of the surrounding network. A resolver should perform these tests when first starting but MAY also perform these tests when it has detected network changes (e.g., address changes, network reattachment, or etc.).

Note: When performing these tests against an address, we make the following assumption about that address: it is a unicast address or an anycast [RFC4786] cluster where all servers have identical configuration and connectivity.

Note: When performing these tests, we also assume that the path is clear of "DNS-interfering" middleboxes, like firewalls, proxies, or forwarders. The presence of such infrastructure can easily make a recursive resolver appear to be functioning improperly. It is beyond the scope of the document as how to work around such interference, although the tests defined in this document may indicate when such misbehaving middleware is causing interference.

Note: This document specifies two sets of tests to perform: a comprehensive one and a fast one. The fast one will detect most common problems; thus, if the fast one passes, then the comprehensive one MAY be considered passed as well.

3.1. Determining DNSSEC Support in Recursive Resolvers

Ideally, a Host Validator can make use of the caching present in recursive resolvers. This section discusses the tests that a recursive resolver MUST pass in order to be fully usable as a DNS cache.

Unless stated otherwise:

- o all of the following tests SHOULD have the Recursion Desired (RD) flag set when sending out a query and queries SHOULD be sent over UDP.

- o the tests MUST NOT have the DNSSEC OK (DO) bit set or utilize any of the other DNSSEC-related requirements, like Extension Mechanisms for DNS (EDNS0).

The tests are designed to check for support of one feature at a time.

3.1.1. Supports UDP Answers

Purpose: This tests basic DNS-over-UDP functionality to a resolver.

Test: A DNS request is sent to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

Note: An implementation MAY chose not to perform the rest of the tests if this test fails, as it is highly unlikely that the resolver under test will pass any of the remaining tests.

3.1.2. Supports TCP Answers

Purpose: This tests basic TCP functionality to a resolver.

Test: A DNS request is sent over TCP to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

3.1.3. Supports EDNS0

Purpose: Test whether a resolver properly supports the EDNS0 extension option.

Prerequisite: Supports UDP or TCP.

Test: Send a request to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com, with an EDNS0 OPT record in the additional section.

SUCCESS: A DNS response was received that contains an EDNS0 option with version number 0.

3.1.4. Supports the DO Bit

Purpose: This tests whether a resolver has minimal support of the DO bit.

Prerequisite: Supports EDNS0.

Test: Send a request to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains the DO bit set.

Note: This only tests that the resolver set the DO bit in the response. Later tests will determine if the DO bit was actually made use of. Some resolvers successfully pass this test because they simply copy the unknown flags into the response. These resolvers will fail the later tests.

3.1.5. Supports the AD Bit DNSKEY Algorithms 5 and/or 8

Purpose: This tests whether the resolver is a validating resolver.

Prerequisite: Supports the DO bit.

Test: Send requests to the resolver under test for an A record for a known existing domain in a DNSSEC-signed zone that is verifiable to a configured trust anchor, such as good-a.test.example.com using the root's published DNSKEY or DS record as a trust anchor. Set the DO bit in the outgoing query. This test should be done twice: once for a zone that contains algorithm 5 (RSASHA1) and again for algorithm 8 (RSASHA256).

SUCCESS: A DNS response was received that contains the Authentic Data (AD) bit set for algorithm 5 (RSASHA1).

BONUS: The AD bit is set for a resolver that supports algorithm 8 (RSASHA256).

3.1.6. Returns RRSIG for Signed Answer

Purpose: This tests whether a resolver will properly return Resource Record Signature (RRSIG) records when the DO bit is set.

Prerequisite: Supports the DO bit.

Test: Send a request to the resolver under test for an A record for a known existing domain in a DNSSEC-signed zone, such as good-a.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains at least one RRSIG record.

3.1.7. Supports Querying for DNSKEY Records

Purpose: This tests whether a resolver can query for and receive a DNSKEY record from a signed zone.

Prerequisite: Supports the DO bit.

Test: Send a request to the resolver under test for a DNSKEY record that is known to exist in a signed zone, such as test.example.com/DNSKEY. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DNSKEY record in the answer section.

Note: Some DNSKEY Resource Record Sets (RRsets) are large and, if the network path has problems with large answers, this query may result in either a false positive or a false negative. In general, the DNSKEY queried for should be small enough to fit into a 1220-byte answer to avoid a false negative result when TCP is disabled. However, querying many zones will result in answers greater than 1220 bytes, so DNS over TCP MUST be available for DNSSEC use in general.

3.1.8. Supports Querying for DS Records

Purpose: This tests whether a resolver can query for and receive a DS record from a signed zone.

Prerequisite: Supports the DO bit.

Test: Send a request to the resolver under test for a DS record that is known to exist in a signed zone, such as test.example.com/DS. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DS record in the answer section.

3.1.9. Supports Negative Answers with NSEC Records

Purpose: This tests whether a resolver properly returns NextSECure (NSEC) records for a nonexisting domain in a DNSSEC-signed zone.

Prerequisite: Supports the DO bit.

Test: Send a request to the resolver under test for an A record that is known to not exist in an NSEC-signed zone, such as nonexistent.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC record.

Note: The query issued in this test MUST be sent to an NSEC-signed zone. Getting back appropriate NSEC3 records does not indicate a failure, but a bad test.

3.1.10. Supports Negative Answers with NSEC3 Records

Purpose: This tests whether a resolver properly returns NSEC3 records ([RFC5155]) for a nonexisting domain in a DNSSEC-signed zone.

Prerequisite: Supports the DO bit.

Test: Send a request to the resolver under test for an A record that is known to be nonexistent in a zone signed using NSEC3, such as nonexistent.nsec3-ns.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC3 record.

Bonus: If the AD bit is set, this validator supports algorithm 7 (RSASHA1-NSEC3-SHA1).

Note: The query issued in this test MUST be sent to an NSEC3-signed zone. Getting back appropriate NSEC records does not indicate a failure, but a bad test.

3.1.11. Supports Queries Where DNAME Records Lead to an Answer

Purpose: This tests whether a resolver can query for an A record in a zone with a known DNAME referral for the record's parent.

Test: Send a request to the resolver under test for an A record that is known to exist in a signed zone within a DNAME-referral child zone, such as good-a.dname-good-ns.test.example.com.

SUCCESS: A DNS response was received that contains a DNAME in the answer section. An RRSIG MUST also be received in the answer section that covers the DNAME record.

3.1.12. Permissive DNSSEC

Purpose: To see if a validating resolver is ignoring DNSSEC validation failures.

Prerequisite: Supports the AD bit.

Test: Ask for data from a broken DNSSEC delegation, such as badsign-a.test.example.com.

SUCCESS: A reply was received with the Rcode set to SERVFAIL.

3.1.13. Supports Unknown RRtypes

Purpose: Some DNS Resolvers/gateways only support some Resource Record Types (RRtypes). This causes problems for applications that need recently defined types.

Prerequisite: Supports UDP or TCP.

Test: Send a request for a recently defined type or an unknown type in the 20000-22000 range, that resolves to a server that will return an answer for all types, such as alltypes.example.com (a server today that supports this is alltypes.res.dnssecready.org).

SUCCESS: A DNS response was retrieved that contains the type requested in the answer section.

3.2. Direct Network Queries

If needed, a Host Validator may need to make direct queries to authoritative servers or known Open Recursive Resolvers in order to collect data. To do that, a number of key network features MUST be functional.

3.2.1. Support for Remote UDP over Port 53

Purpose: This tests basic UDP functionality to outside the local network.

Test: A DNS request is sent to a known distant authoritative server for a record known to be within that server's authoritative data. Example: send a query to the address of ns1.test.example.com for the good-a.test.example.com/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: An implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but it does not need to be.

3.2.2. Support for Remote UDP with Fragmentation

Purpose: This tests if the local network can receive fragmented UDP answers.

Prerequisite: Local UDP traffic > 1500 bytes in size is possible.

Test: A DNS request is sent over UDP to a known distant DNS address asking for a record that has an answer larger than 2000 bytes. For example, send a query for the test.example.com/DNSKEY record with the DO bit set in the outgoing query.

SUCCESS: A DNS response was received that contains the large answer.

Note: A failure in getting large answers over UDP is not a serious problem if TCP is working.

3.2.3. Support for Outbound TCP over Port 53

Purpose: This tests basic TCP functionality to outside the local network.

Test: A DNS request is sent over TCP to a known distant authoritative server for a record known to be within that server's authoritative data. Example: send a query to the address of ns1.test.example.com for the good-a.test.example.com/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: An implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but it does not need to be.

3.3. Support for DNSKEY and DS Combinations

Purpose: This test can check what algorithm combinations are supported.

Prerequisite: Supports the AD bit for Algorithms 5 and/or 8.

Test: A DNS request is sent over UDP to the resolver under test for a known combination of the DS algorithm number (N) and DNSKEY algorithm number (M) of the example form ds-N.alg-M-nsec.test.example.com. Examples:

```
ds-2.alg-13-nsec.test.example.com TXT
     or
ds-4.alg-13-nsec3.test.example.com TXT
```

SUCCESS: A DNS response is received with the AD bit set and with a matching record type in the answer section.

Note: For algorithms 6 and 7, NSEC is not defined; thus, a query for alg-M-nsec3 is required. Similarly, NSEC3 is not defined for algorithms 1, 3, and 5. Furthermore, algorithms 2, 4, 9, and 11 do not currently have definitions for signed zones.

4. Aggregating the Results

Some conclusions can be drawn from the results of the above tests in an "aggregated" form. This section defines some labels to assign to a resolver under test given the results of the tests run against them.

4.1. Resolver Capability Description

This section will group and label certain common results.

Resolvers are classified into the following broad behaviors:

Validator: The resolver passes all DNSSEC tests and had the AD bit appropriately set.

DNSSEC-Aware: The resolver passes all DNSSEC tests, but it does not appropriately set the AD bit on answers, indicating it is not validating. A Host Validator will function fine using this resolver as a forwarder.

Non-DNSSEC-Capable: The resolver is not DNSSEC-aware and will make it hard for a Host Validator to operate behind it. It MAY be usable to query for data that is in known insecure sections of the DNS tree.

Not a DNS Resolver: This is an improperly behaving resolver and should not be used at all.

While it would be great if all resolvers fell cleanly into one of the broad categories above, that is not the case. For that reason, it is necessary to augment the classification with a more descriptive result. This is done by adding the word "Partial" in front of Validator/DNSSEC-aware classifications, followed by sub-descriptors of what is not working.

Unknown: Failed the unknown test

DNAME: Failed the DNAME test

NSEC3: Failed the NSEC3 test

TCP: TCP not available

SlowBig: UDP is size limited, but TCP fallback works

NoBig: TCP not available and UDP is size limited

Permissive: Passes data known to fail validation

5. Roadblock Avoidance

The goal of this document is to tie the above tests and aggregations to avoidance practices; however, the document does not specify exactly how to do that.

Once we have determined what level of support is available in the network, we can determine what must be done in order to effectively act as a validating resolver. This section discusses some of the options available given the results from the previous sections.

The general fallback approach can be described by the following sequence:

> If the resolver is labeled as "Validator" or "DNSSEC-aware":
>
> > Send queries through this resolver and perform local validation on the results.
> >
> > If validation fails, try the next resolver.
>
> Else, if the resolver is labeled "Not a DNS Resolver" or "Non-DNSSEC-capable":
>
> > Mark it as unusable and try the next resolver.

Else if no more resolvers are configured and if direct queries are supported:

1. Try iterating from the Root.

2. If the answer is SECURE/BOGUS:
   Return the result of the iteration.

3. If the answer is INSECURE:
   Re-query "Non-DNSSEC-capable" servers and return answers from them without the AD bit set to the client.

This will increase the likelihood that split-view unsigned answers are found.

Else:

Return an error code and log a failure.

While attempting resolution through a particular recursive name server with a particular transport method that worked, any transport-specific parameters MUST be remembered in order to avoid any unnecessary fallback attempts.

Transport-specific parameters MUST also be remembered for each authoritative name server that is queried while performing an iterative mode lookup.

Any transport settings that are remembered for a particular name server MUST be periodically refreshed; they should also be refreshed when an error is encountered as described below.

For a stub resolver, problems with the name server can manifest themselves under the following types of error conditions:

- No Response, error response, or missing DNSSEC metadata

- Illegal Response: An illegal response is received, which prevents the validator from fetching all the necessary records required for constructing an authentication chain. This could result when referral loops are encountered, when any of the antecedent zone delegations are lame, when aliases are erroneously followed for certain RRtypes (such as Start of Authority (SOA), DNSKEYs, or DS records), or when resource records for certain types (e.g., DS) are returned from a zone that is not authoritative for such records.

- o Bogus Response: A Bogus Response is received, when the cryptographic assertions in the authentication chain do not validate properly.

For each of the above error conditions, a validator MAY adopt the following dynamic fallback technique, preferring a particular approach if it is known to work for a given name server or zone from previous attempts.

- o No response, error response, or missing DNSSEC metadata:

  * Retry with different EDNS0 sizes (4096, 1492, or None).

  * Retry with TCP only.

  * Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.

  * Retry using an alternative transport method, if this alternative method is known (configured) to be supported by the name server in question.

- o Illegal Response

  * Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.

  * Check if any of the antecedent zones up to the closest configured trust anchor are Insecure.

- o Bogus Response

  * Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.

For each fallback technique, attempts to reach multiple potential name servers should be skewed such that the next name server is tried when the previous one returns an error or a timeout is reached.

The validator SHOULD remember, in its zone-specific fallback cache, any broken behavior identified for a particular zone for a duration of that zone’s SOA-negative TTL.

The validator MAY place name servers that exhibit broken behavior into a blacklist and bypass these name servers for all zones that they are authoritative for. The validator MUST time out entries in this name server blacklist periodically, where this interval could be set to be the same as the DNSSEC BAD cache default TTL.

5.1. Partial Resolver Usage

It may be possible to use Non-DNSSEC-Capable caching resolvers in careful ways if maximum optimization is desired. This section describes some of the advanced techniques that could be implemented to use a resolver in at least a minimal way. Most of the time, this would be unnecessary; the exception being the case where none of the resolvers are fully compliant and, thus, the choice would be to use them at least minimally or not at all (and no caching benefits would be available).

5.1.1. Known Insecure Lookups

If a resolver is Non-DNSSEC-Capable but a section of the DNS tree has been determined to be Insecure [RFC4035], then queries to this section of the tree MAY be sent through the Non-DNSSEC-Capable caching resolver.

5.1.2. Partial NSEC/NSEC3 Support

Resolvers that understand DNSSEC generally, and understand NSEC but not NSEC3, are partially usable. These resolvers generally also lack support for unknown types, rendering them mostly useless and to be avoided.

6. Start-Up and Network Connectivity Issues

A number of scenarios will produce either short-term or long-term connectivity issues with respect to DNSSEC validation. Consider the following cases:

Time Synchronization: Time synchronization problems can occur when a device has been off for a period of time and the clock is no longer in close synchronization with "real time" or when a device always has the clock set to the same time during start-up. This will cause problems when the device needs to resolve its source of time synchronization, such as "ntp.example.com".

Changing Network Properties: A newly established network connection may change state shortly after an HTTP-based paywall authentication system has been used. This is especially common in hotel, airport, and coffee-shop networks where DNSSEC, validation, and even DNS are not functional until the user proceeds through a series of forced web pages used to enable their network. The tests in Section 3 will produce very different results before and after the network authorization has succeeded. APIs exist on many operating systems to detect initial network device status changes, such as right after DHCP has finished, but few (none?) exist to detect that authentication through a paywall has succeeded.

There are only two choices when situations like this happen:

Continue to perform DNSSEC processing, which will likely result in all DNS requests failing. This is the most secure route, but causes the most operational grief for users.

Turn off DNSSEC support until the network proves to be usable. This allows the user to continue using the network, at the cost of security. It also allows for a denial-of-service attack if a man-in-the-middle can convince a device that DNSSEC is impossible.

6.1. What to Do

If the Host Validator detects that DNSSEC resolution is not possible, it SHOULD log the event and/or SHOULD report an error to the user. In the case where there is no user, no reporting can be performed; thus, the device MAY have a policy of action, like continue or fail. Until middleboxes allow DNSSEC-protected information to traverse them consistently, software implementations may need to offer this choice to let users pick the security level they require. Note that continuing without DNSSEC protection in the absence of a notification or report could lead to situations where users assume a level of security that does not exist.

7. Quick Test

The quick tests defined below make the assumption that the questions to be asked are of a real resolver; and the only real question is: "How complete is the DNSSEC support?". This quick test has been implemented in a few programs developed at IETF hackathons at IETF 93 and IETF 94. The programs use a common grading method. For each question that returns an expected answer, the resolver gets a point. If the AD bit is set as expected, the resolver gets a second point.

7.1. Test Negative Answers Algorithm 5

Query: realy-doesnotexist.test.example.com. A

Answer: RCODE= NXDOMAIN, Empty Answer, Authority: NSEC-proof

7.2. Test Algorithm 8

Query: alg-8-nsec3.test.example.com. SOA

Answer: RCODE= 0, Answer: SOA record

7.3. Test Algorithm 13

Query: alg-13-nsec.test.example.com. SOA

Answer: RCODE= 0, Answer: SOA record

7.4. Fails When DNSSEC Does Not Validate

Query: dnssec-failed.test.example.com. SOA

Answer: RCODE= SERVFAIL, empty answer, and authority, AD=0

8. Security Considerations

This document discusses problems that may occur while deploying the DNSSEC protocol. It describes what may be possible to help detect and mitigate these problems. Following the outlined suggestions will result in a more secure DNSSEC-operational environment than if DNSSEC was simply disabled.

9. Normative References

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

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

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

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

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

Acknowledgments

We thank the IESG and DNSOP working group members for their extensive comments and suggestions.

Authors' Addresses

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
United States of America

Email: ietf@hardakers.net

Olafur Gudmundsson
CloudFlare
San Francisco, CA 94107
United States of America

Email: olafur+ietf@cloudflare.com

Suresh Krishnaswamy
Parsons
7110 Samuel Morse Dr
Columbia, MD 21046
United States of America

Email: suresh@tislabs.com